Title
Precomputing strategy for Hamiltonian Monte Carlo method based on regularity in parameter space

Journal
Computational Statistics, 32(1)

ISSN
0943-4062

Authors
Zhang, C
Shahbaba, B
Zhao, H

Publication Date
2017-03-01

DOI
10.1007/s00180-016-0683-1

Peer reviewed

PRECOMPUTING STRATEGY FOR HAMILTONIAN MONTE CARLO METHOD BASED ON REGULARITY IN PARAMETER SPACE

CHENG ZHANG*, BABAK SHAHBABA†, AND HONGKAI ZHAO*

Abstract. Markov Chain Monte Carlo (MCMC) algorithms play an important role in statistical inference problems dealing with intractable probability distributions. Recently, many MCMC algorithms such as Hamiltonian Monte Carlo (HMC) and Riemannian Manifold HMC have been proposed to provide distant proposals with high acceptance rate. These algorithms, however, tend to be computationally intensive which could limit their usefulness, especially for big data problems due to repetitive evaluations of functions and statistical quantities that depend on the data. This issue occurs in many statistic computing problems. In this paper, we propose a novel strategy that exploits smoothness (regularity) in parameter space to improve computational efficiency of MCMC algorithms. When evaluation of functions or statistical quantities are needed at a point in parameter space, interpolation from precomputed values or previous computed values is used. More specifically, we focus on Hamiltonian Monte Carlo (HMC) algorithms that use geometric information for faster exploration of probability distributions. Our proposed method is based on precomputing the required geometric information on a set of grids before running sampling algorithm and approximating the geometric information for the current location of the sampler using the precomputed information at nearby grids at each iteration of HMC. Sparse grid interpolation method is used for high dimensional problems. Tests on computational examples are shown to illustrate the advantages of our method.

Key words. Hamiltonian Monte Carlo; Force map; Sparse grid interpolation; Precomputing

1. Introduction. Many statistical and machine learning methods rely on costly iterative algorithms for optimization or sampling. One of the most computationally intensive components of these methods is repetitive evaluations of functions, their derivatives, geometric and statistical quantities that depend on the data. This is especially challenging in Big Data problems. To reduce the computation cost, one common approach is subsampling, which restricts the computation to a subset of the data or sample, such as stochastic gradient methods (See, for example, [17, 27]). Another approach could be to find some computationally cheaper surrogate functions to substitute the expensive objective functions. (See, for example, [24, 25].) In this paper, we propose a different approach that explores smoothness or regularity in parameter space, which is true for most statistical models. When evaluation of functions or statistics quantities are needed at a point in parameter space, interpolation from precomputed values or previous computed values is used. Here, we mainly focus on a state-of-the-art class of Markov Chain and Monte Carlo (MCMC) sampling algorithms called Hamiltonian Monte Carlo (HMC). However, our proposed method could be extended to other computationally intensive, iterative algorithms commonly used in statistics and machine learning.

MCMC was first introduced by Metropolis [20] to simulate the distribution of states for a system of idealized molecules. Almost contemporarily, Alder and Wainwright [4] proposed a deterministic approach to molecules simulation called molecular dynamics (MD). In the following decades, the MCMC and molecular dynamics approaches have continued to develop in their respective areas. In 1987, Duane, Kennedy, Pendleton and Roweth [13] made a remarkable breakthrough by developing a Hybrid Monte Carlo (HMC) algorithm based on combining MCMC and molecular dynamics approaches. This is also known as Hamiltonian Monte Carlo (HMC) in the literature. Neal [22] provided an extensive review of this method and presented several extensions. The basic idea is that starting from the current state of MCMC, one can use MD to generate trial moves (i.e., proposals within the Metropolis algorithm) that can move far from the current state (resulting

[1]Department of Mathematics, University of California, Irvine, USA
[2]Department of Statistics, University of California, Irvine, USA

in low autocorrelations) while keeping the acceptance probability high (by moving towards high probability regions). Therefore, the HMC sampling method can provide more rapid and efficient exploration of the parameter space than standard random walk proposals. However, HMC requires expensive gradient computations in order to simulate the Hamiltonian dynamics system. This could be infeasible in Big Data problems. Therefore, in recent years, there have been many attempts to improve computational efficiency of HMC and its variants. (See for example, [27, 1, 15, 16, 25, 6, 9, 19, 2, 3].) One possible strategy is to use small subsets of data at each iteration [27, 25]. As an alternative approach, the precomputing strategies we propose here can reduce the computation cost of HMC while maintaining the overall efficiency of the method by exploiting smooth dependence of parameters that exists in typical statistical models.

Before we present our method, we first briefly review HMC in the following section. We then present our proposed method in Section 3 and evaluate its performance in Section 4 using several examples. In Section 5, we discuss an extension of our method that is faster computationally, but converges to an approximation of the target distribution. Finally, Section 6 is devoted to discussion of future research directions and applications of our method in other algorithms.

2. Hamiltonian Monte Carlo. In Bayesian Statistics, we are interested in sampling from the posterior distribution of the model parameters q given the observed data, $Y = (y_1, y_2, \ldots, y_N)^T$,

$$P(q|Y) \propto \exp(-U(q)), \tag{2.1}$$

where the potential energy function U is defined as

$$U(q) = -\sum_{i=1}^{N} \log P(y_i|q) - \log P(q). \tag{2.2}$$

Here, the first term is the negative log-likelihood, and $P(q)$ is the assumed prior on model parameters. The posterior distribution is almost always intractable. Therefore, Markov Chain Monte Carlo (MCMC) algorithms are typically used for sampling from the posterior distribution to perform statistical inference. We could for example use the Metropolis algorithm as follows. Given the current state, q, we propose a new state, q^* using a symmetric proposal distribution such that $P(q^*|q) = P(q|q^*)$. We then accept the proposed state as our new state with the following probability:

$$\min(1, \exp[U(q) - U(q^*)])$$

The standard random walk Metropolis generates proposals by sampling from a normal distribution with its mean set to the current state, q. There are more efficient strategies to generate proposals. Among these, Hamiltonian Monte Carlo (HMC) has become increasingly popular due to its capability of making distant proposals (i.e., low autocorrelation) with high acceptance probability. More specifically, HMC introduces a Hamiltonian dynamics system with auxiliary momentum variables p to propose samples of q in a Metropolis framework that explores the parameter space more efficiently compared to standard random walk proposals. Following the dynamics of the introduced Hamiltonian system, HMC generates proposals jointly for (q, p) using the following system of differential equations:

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i} \tag{2.3}$$

$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i} \tag{2.4}$$

Algorithm 1: Hamiltonian Monte Carlo

Input: Starting position $q^{(1)}$ and step size ϵ
for $t = 1, 2, \cdots$ **do**
  Resample momentum p
  $p^{(t)} \sim \mathcal{N}(0, M),\ (q_0, p_0) = (q^{(t)}, p^{(t)})$
  Simulate discretization of Hamiltonian dynamics:
  for $l = 1$ *to* L **do**
    $p_{l-1} \leftarrow p_{l-1} - \frac{\epsilon}{2}\frac{\partial U}{\partial q}(q_{l-1})$
    $q_l \leftarrow q_{l-1} + \epsilon M^{-1} p_{l-1}$
    $p_l \leftarrow p_l - \frac{\epsilon}{2}\frac{\partial U}{\partial q}(q_l)$
  $(q^*, p^*) = (q_L, p_L)$
  Metropolis-Hasting correction:
  $u \sim \text{Uniform}[0, 1]$
  $\rho = \exp[H(q^{(t)}, p^{(t)}) - H(q^*, p^*)]$
  if $u < \min(1, \rho)$, **then** $q^{(t+1)} = q^*$

where the Hamiltonian function is defined as $H(q, p) = U(q) + \frac{1}{2}p^T M^{-1} p$. The quadratic kinetic energy function $K(p) = \frac{1}{2}p^T M^{-1} p$ corresponds to the negative log-density of a zero-mean multivariate Gaussian distribution with the covariance M. Here, M is known as the mass matrix, which is often set to the identity matrix, I, but can be used to precondition the sampler using Fisher information [15]. By simulating the Hamiltonian dynamics system together with the correction (i.e., accept/reject) step, HMC generates samples from a joint distribution of (q, p) defined by

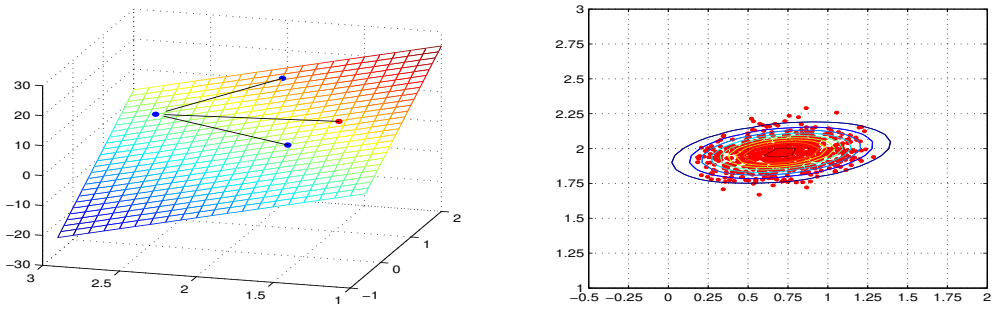
$$P(q, p) \propto \exp\left(-U(q) - \frac{1}{2}p^T M^{-1} p\right)$$

Notice that q and p are independent in general.

Each sample from the HMC algorithm is generated by two steps: the proposal step and the correction step. In the proposal step, new values for the momentum variable p are drawn from their Gaussian distribution. Starting from the current state (q, p), the Hamiltonian dynamics system (2.3),(2.4) is simulated for L steps using the leapfrog method (Algorithm 1), with a stepsize of ϵ. Here, L and ϵ are parameters which needs to be tuned to obtain a reasonable acceptance probability. In the correction step, the proposed state (q^*, p^*) at the end of the trajectory is accepted as the next state of the Markov chain with probability $\min(1, \exp[-H(q^*, p^*) + H(q, p)])$ and the position variable q is updated correspondingly. These steps are presented in Algorithm 1.

Note that in Algorithm 1, when simulating the Hamiltonian dynamics system, we need to repeatedly compute the gradient of the potential energy function U. This could be extremely time consuming. Many attempts have been made in recent years to reduce this cost in order to improve the overall computational efficiency of HMC. See for example, Shahbaba et al. [25] and Chen [10]. In this paper, we claim that the smooth dependence on parameters can be exploited to approximate the gradient effectively using precomputed or previous-computed values on a grid of points. We will discuss our method in details in the following section.

3. Precomputing Strategies.



(a) $\nabla_\mu U$ (b) HMC samples

FIG. 3.1. *2D Gaussian example: (a) the graph of the first component of $\nabla_\mu U$. The function value at red point can be obtained by interpolation when the function values at three blue points are known. (b) HMC samples from the posterior distribution.*

3.1. Insights from an Illustrative Example. We start with a simple example to motivate our approach. Consider a bivariate Gaussian distribution with known covariance matrix and a conjugate prior

$$Y|\mu \sim \mathcal{N}(\mu, \Sigma), \quad \mu \sim \mathcal{N}(\mu_0, \Sigma_0)$$

Note that in this case, the posterior distribution has a closed form so MCMC is not required. However, we use this example to motivate our method. For this problem, the potential energy function and its gradient are given by

$$U(\mu) = \frac{1}{2}\sum_{i=1}^{N}(y_n - \mu)^T\Sigma^{-1}(y_n - \mu) + \frac{1}{2}(\mu - \mu_0)^T\Sigma_0^{-1}(\mu - \mu_0) \tag{3.1}$$

$$\frac{\partial U}{\partial \mu} = N\Sigma^{-1}(\mu - \bar{Y}) + \Sigma_0^{-1}(\mu - \mu_0), \quad \bar{Y} = \sum_{i=1}^{N} y_i/N \tag{3.2}$$

In the gradient function (3.2), all the information about the parameter is contained in one single value $\bar{Y}$ (i.e., the *sufficient statistic* for μ). Therefore, if we precompute $\bar{Y}$, gradient computation of the potential energy function U could be reduced to a simple matrix vector multiplication. Moreover, the gradient function itself is a linear function. In this 2D case, the essential information of $\frac{\partial U}{\partial \mu}$ can be captured by its function values at three non-collinear points (left panel of Fig.3.1). On the other hand, samples from the posterior distribution are concentrated around the high density region where the neighborhood of one sample is frequently visited in the simulations of Hamiltonian dynamics (right panel of Fig.3.1). We use these insights to develop a method that can approximates the gradient function using precomputed values in order to accelerate standard HMC.

3.2. Force Approximation. If we could solve Hamilton's equations (2.3 and 2.4) analytically, the acceptance probability of new proposals in HMC would be exactly one (i.e., each proposal is accepted) because of the conservation of the Hamiltonian [22]. However, since solving these equations exactly is too hard in practice, we usually approximate them by discretizing time and using the leapfrog method (Algorithm 1). As a result, the acceptance probability may be less than

one. The tradeoff between the accuracy of the proposal-generating mechanisms and Metropolis acceptance probability can go beyond time discretization. Therefore, in this paper we ask the following question: *can we properly approximate the proposal-generating mechanism in order to reduce computational complexity while keeping the acceptance probability at a reasonable level?* We answer this question in the remaining part of this section.

Note that we can rewrite Hamilton's equations as follows:

$$\frac{dq_i}{dt} = [M^{-1}p]_i \tag{3.3}$$

$$\frac{dp_i}{dt} = -\frac{\partial U}{\partial q_i} \tag{3.4}$$

The routine of the trajectory for one proposal step will be determined by both the random initialization of the momentum and the negative gradient of the potential energy function, which is called *force* in Physics,

$$F = -\frac{\partial U}{\partial q}$$

The random momentum enables the scheme to explore the target distribution stochastically, and the fictitious force guides the sampler in the right direction so that the entire sampling method would be more efficient than random walk proposals. However, the computation of the true force F is quite expensive. To alleviate this issue, we propose to construct a Hamiltonian dynamics system, at this time for the *proposal step* only, using an alternative Hamiltonian function,

$$\tilde{H}(q,p) = \tilde{U}(q) + K(p)$$

where $\tilde{U}$ is an approximation to the potential energy U, whose corresponding negative gradient $\tilde{F}$ (which is an approximation to the true force function F) can be computed relatively fast. Since the simulation of a Hamiltonian dynamics system only involves the force function, it suffices to find an approximate force function, $\tilde{F}$, directly.

3.3. Naive Grid HMC. To approximate the force function, one could simply use a piecewise constant functions, which corresponds to a piecewise linear approximation of the potential energy. In most cases, the high density region of the posterior distribution can be covered by a finite domain D, henceforth called "domain of interest". If we partition D with a fine grid, justification of an appropriate piecewise constant approximation to the force function F is guaranteed by the smooth dependence of F (or U) in parameter space. For a 2-dimensional problem, suppose our domain of interest is $D = [a,b] \times [c,d]$. Given the grid points

$$x_i = a + i\Delta x, \quad y_j = c + j\Delta y, \quad i,j = 0,1,\ldots,N_p$$

where Δx, Δy are the corresponding grid sizes, for each cell, $C_{i,j} = [x_{i-1}, x_i] \times [y_{j-1}, y_j]$, we approximate the force function by its value at the center of the grid, $c_{i,j} = (x_{i-1/2}, y_{j-1/2})$:

$$\tilde{F}(q) = F_{i,j} \stackrel{\triangle}{=} F(c_{i,j}), \text{ if } q \in C_{i,j}$$

By the smoothness of F, $\|\tilde{F} - F\|_\infty \to 0$ as $\Delta x, \Delta y \to 0$. Therefore, we can always find some fine grid to achieve the desired approximation accuracy.

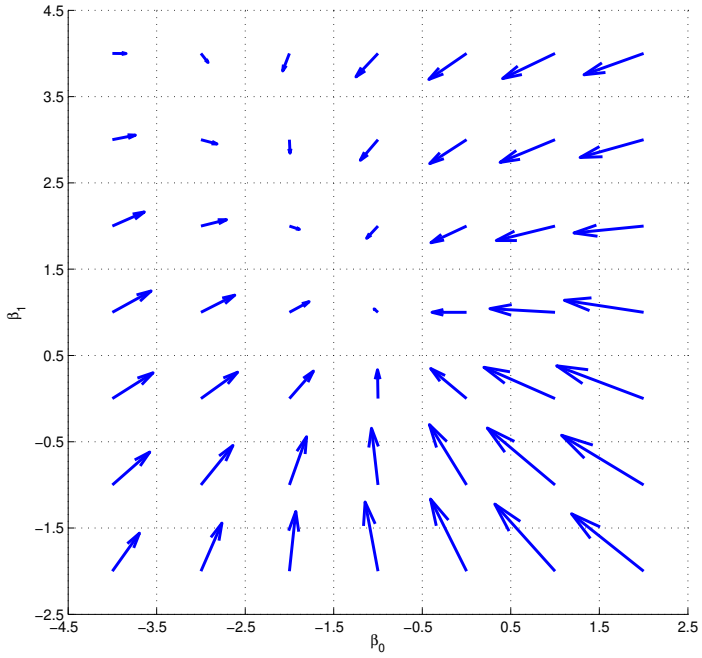


FIG. 3.2. *Force map of a logistic regression model*

Figure 3.2 shows a piecewise constant approximation to the force function of a logistic regression model with design matrix $X = (\mathbf{1}, X_1)$ and true parameter $\beta = (-1, 1)^T$, where X_1 follows standard normal distribution. The binary responses $Y = (y_1, y_2, \ldots, y_N)^T$ are sampled independently from Bernoulli distributions

$$y_i \sim \text{Bernoulli}(p_i), \quad p_i = \frac{\exp(x_i\beta)}{1 + \exp(x_i\beta)}$$

Therefore, the likelihood function is

$$L(\beta|X, Y) \propto \prod_{i=1}^{N} p_i^{y_i}(1 - p_i)^{1-y_i}$$

and the potential energy function and the force function are

$$U(\beta) = -\sum_{i=1}^{N} \left[y_i x_i \beta - \log(1 + \exp(x_i\beta))\right] \tag{3.5}$$

$$F(\beta) = -\frac{\partial U}{\partial \beta} = X^T(Y - P) \tag{3.6}$$

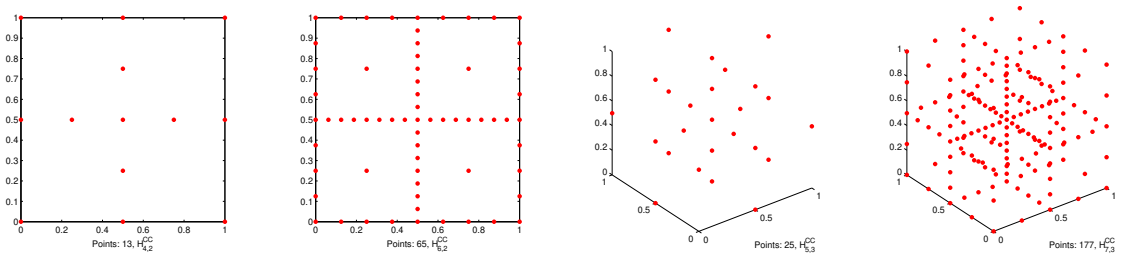
Algorithm 2: Naive Grid HMC

Input: Starting position $q^{(1)}$ and step size ϵ
Precompute the approximate force map $\tilde{F} : F_{i,j} = -\left(\frac{\partial U}{\partial q}\right)_{i,j}$
for $t = 1, 2, \cdots$ **do**
  Resample momentum p
  $p^{(t)} \sim \mathcal{N}(0, M)$, $(q_0, p_0) = (q^{(t)}, p^{(t)})$
  Simulate discretization of Hamiltonian dynamics:
  Find the position of q_0 in the force map: (i_0, j_0)
  for $l = 1$ *to* L **do**
    $p_{l-1} \leftarrow p_{l-1} + \frac{\epsilon}{2} F_{i_{l-1}, j_{l-1}}$
    $q_l \leftarrow q_{l-1} + \epsilon M^{-1} p_{l-1}$
    Find the position of q_l in the force map: (i_l, j_l)
    $p_l \leftarrow p_l + \frac{\epsilon}{2} F_{i_l, j_l}$
  $(q^*, p^*) = (q_L, p_L)$
  Metropolis-Hasting correction:
  $u \sim$ Uniform$[0, 1]$
  $\rho = e^{H(q^{(t)}, p^{(t)}) - H(q^*, p^*)}$
  if $u < \min(1, \rho)$, **then** $q^{(t+1)} = q^*$

where $P = (p_1, p_2, \ldots, p_N)^T$. It can be seen from the graph that: (i) the approximate force "map" does point to the right direction so that it provides valid geometric information for HMC; (ii) the approximate force function also changes smoothly, which means that the numerical stability of the leap-frog scheme can be maintained with approximately the same step size as standard HMC. As a result, the proposed scheme with piecewise constant force functions would be consistent and stable. Therefore, we can precompute the piecewise constant function $\tilde{F}$ in advance. When evaluating the force function in the simulation of the Hamiltonian dynamics system, we locate the cell (i, j) for the current parameter q and read the approximate function value $\tilde{F}(q)$ from the precomputed force map. We summarize this approach in Algorithm 2 and refer to it as Grid HMC (GHMC).

Our initial results showed that the Naive Grid HMC method would work well for simple problems. However, implementation of this method in general involves two challenges. First, its extension to high dimensional problems could be problematic because as the number of parameters increases, the number of grid nodes at which we need to evaluate the approximate force map grows exponentially. In other words, the method will encounter the curse of dimensionality. The second challenge is related to finding the domain of interest. We will address those two issues in the following two subsections respectively.

3.4. Sparse Grid HMC. The sparse grid interpolation method use a special discretization technique to approximate a smooth function over a sparse grid of points ([8, 18, 5]). More specifically, it uses a hierarchical basis (a representation of a discrete function space that is equivalent to the conventional nodal basis) and a sparse tensor product construction. Discretization on sparse grids employs $\mathcal{O}(N \cdot \log(N)^{d-1})$ grid points only, where d denotes the dimension and N denotes the number of grid points at the boundary in each coordinate direction (i.e., the mesh size is $h = 1/N$). Using piecewise linear basis functions, the interpolation accuracy could be $\mathcal{O}(N^{-2} \cdot (\log N)^{d-1})$ with



$H^{CC}_{4,2}$ $H^{CC}_{6,2}$ $H^{CC}_{5,3}$ $H^{CC}_{7,3}$

FIG. 3.3. *Clenshaw-Curtis type sparse grids*

respect to the L_2 norm and L_∞ norm if the solution has bounded second mixed derivatives. Note that for a full grid, N^d grid points are needed to achieve an approximation accuracy of $\mathcal{O}(N^{-2})$.

3.4.1. Smolyak's formula. Assume that we want to approximate the smooth functions $f : [0,1]^d \to \mathbb{R}$ using a finite number of function values (at support nodes). For the one dimensional case, the interpolation formula is given by

$$U^i(f) = \sum_{j=1}^{m_i} f(x_j^i) \cdot a_j^i$$

where $i \in \mathbb{N}$, $X^i = \{x_j^i \in [0,1] | j = 1, \ldots, m_i\}$ are the support nodes, and $a_j^i \in C([0,1])$ are the basis functions. We could use the following tensor product for multidimensional cases:

$$(U^{i_1} \otimes \cdots \otimes U^{i_d}) = \sum_{j_1=1}^{m_{i_1}} \cdots \sum_{j_d=1}^{m_{i_d}} f(x_{j_1}^{i_1}, \ldots, x_{j_d}^{i_d}) \cdot (a_{j_1}^{i_1} \otimes \cdots \otimes a_{j_d}^{i_d}) \tag{3.7}$$

However, the above product formula requires a large number $(m_{i_1} \cdots m_{i_d})$ of support nodes, which are sampled on the full grid. Smolyak's formula then can be applied here to reduce the number of support nodes while maintaining the approximation quality of the interpolation formula up to a logarithmic factor. With $U^0 = 0$, define

$$\Delta^i = U^i - U^{i-1}, \quad \forall\, i \in \mathbb{N}$$

Moreover, we put $|\boldsymbol{i}| = i_1 + \cdots + i_d$ for $\boldsymbol{i} \in \mathbb{N}^d$. Then Smolyak's algorithm is given by

$$A_{q,d}(f) = \sum_{|\boldsymbol{i}| \le q} (\Delta^{i_1} \otimes \cdots \otimes \Delta^{i_d})(f) = A_{q-1,d}(f) + \underbrace{\sum_{|\boldsymbol{i}| = q} (\Delta^{i_1} \otimes \cdots \otimes \Delta^{i_d})(f)}_{\Delta A_{q,d}(f)} \tag{3.8}$$

for integers $q \ge d$, where $A_{d-1,d} = 0$. In fact, (3.8) can be presented in terms of the univariate interpolation formulas [26],

$$A_{q,d}(f) = \sum_{q-d+1 \le |\boldsymbol{i}| \le q} (-1)^{q-|\boldsymbol{i}|} \cdot \binom{d-1}{q-|\boldsymbol{i}|} \cdot (U^{i_1} \otimes \cdots \otimes U^{i_d})(f)$$

Algorithm 3: Sparse Grid HMC

Input: Starting position $q^{(1)}$ and step size ϵ
Precompute the hierarchical surpluses for Smolyak's formula $A_{k+d,d}$ of potential energy U
for $t = 1, 2, \cdots$ **do**
  Resample momentum p
  $p^{(t)} \sim \mathcal{N}(0, M)$
  $(q_0, p_0) = (q^{(t)}, p^{(t)})$
  Simulate discretization of Hamiltonian dynamics
  for $l = 1$ *to* L **do**
    $p_{l-1} \leftarrow p_{l-1} - \frac{\epsilon}{2}\nabla A_{k+d,d}(U)(q_{l-1})$
    $q_l \leftarrow q_{l-1} + \epsilon M^{-1} p_{l-1}$
    $p_l \leftarrow p_l - \frac{\epsilon}{2}\nabla A_{k+d,d}(U)(q_l)$
  $(q^*, p^*) = (q_L, p_L)$
  Metropolis-Hasting correction:
  $u \sim \text{Uniform}[0, 1]$
  $\rho = e^{H(q^{(t)}, p^{(t)}) - H(q^*, p^*)}$
  if $u < \min(1, \rho)$, **then** $q^{(t+1)} = q^*$

Therefore, only the function values at the sparse grid

$$H_{q,d} = \bigcup_{q-d+1 \le |\boldsymbol{i}| \le q} (X^{i_1} \times \cdots \times X^{i_d}) \tag{3.9}$$

are needed to evaluate $A_{q,d}(f)$. It is better to select the sets X^i in a nested fashion ($X^i \subset X^{i+1}$) to obtain many recurring points with increasing q.

3.4.2. Sparse grid and Multivariate hierarchical structure. There are many possibilities to construct nested sparse grids. As an example, Fig. 3.3 shows the Clenshaw-Curtis type sparse grids H^{CC} in two and three dimensional spaces. With appropriate sparse grid and basis functions a, the multivariate interpolation formula (3.8) can be implemented in a hierarchical form where

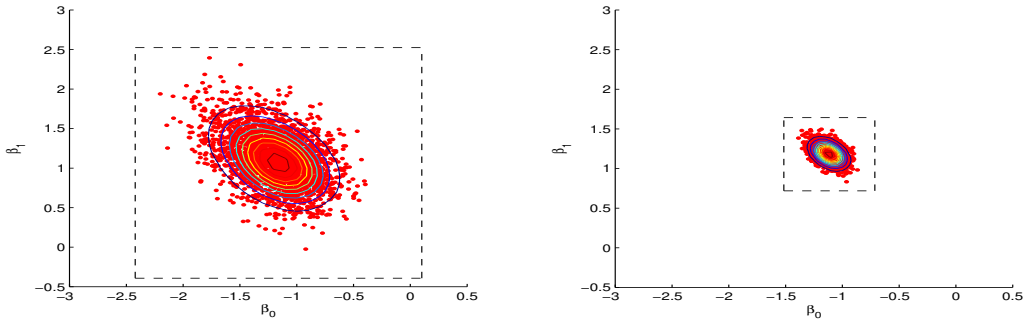
$$\Delta A_{q,d}(f) = \sum_{|\boldsymbol{i}|=q} \sum_{x_{j_1}^{i_1} \in X_\Delta^{i_1}} \cdots \sum_{x_{j_d}^{i_d} \in X_\Delta^{i_d}} \left(f(x_{j_1}^{i_1}, \ldots, x_{j_d}^{i_d}) - A_{q-1,d}(f)(x_{j_1}^{i_1}, \ldots, x_{j_d})\right) \cdot (a_{j_1}^{i_1} \otimes \cdots \otimes a_{j_d}^{i_d}) \tag{3.10}$$

The hierarchical surpluses

$$w_{\boldsymbol{j}}^{k,\boldsymbol{i}} \triangleq f(\boldsymbol{x}_{\boldsymbol{j}}^{\boldsymbol{i}}) - A_{k+d-1,d}(\boldsymbol{x}_{\boldsymbol{j}}^{\boldsymbol{i}})$$

introduced by Bungartz [7] can be used to obtain an estimate of the current approximation error and terminate the algorithm automatically when a desired accuracy is reached. More detailed information about the construction of sparse grid and basis functions and the derivation of the hierarchical form is provided in the appendix.

Using sparse grid interpolation (3.8) based on the Smolyak algorithm, we can generalize our Naive Grid HMC method to relatively higher dimensional problems. The hierarchical surpluses for



(a) $N = 100$ (b) $N = 1000$

FIG. 3.4. *Domains of interest using Laplace's approximation for the logistic regression model.*

the energy function U can be precomputed with certain type of sparse grid and $\nabla A_{q,d}$ can be called to replace the gradient computation (see Algorithm 3).

3.5. Domain of Interest. The grid needs to be specified over a finite domain of interest such that there is a good balance between the cost and efficiency of the precomputing strategy. That is, we need to find an appropriate bounded domain that covers most of the high density region without creating cells that are rarely visited by the sampler. Note that for points outside of this domain of interest, one can still use the standard HMC method; however, there will not be any computational saving for these points.

To find the domain of interest, we use Laplace's approximation,

$$q|Y \dot{\sim} \mathcal{N}(\hat{q}, \mathcal{J}^{-1}(\hat{q}))$$

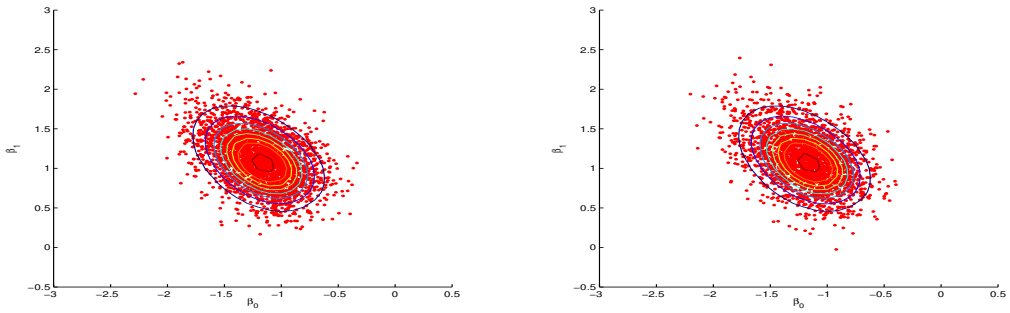
where $\hat{q}$ is the posterior mode which can be estimated using fast optimization methods, and $\mathcal{J}(\hat{q}) = H_U(\hat{q})$ is the Hessian matrix at the point. Given a pre-specified probability, p, we can find a domain with probability p based on the above normal approximation.

Figure 3.4 shows the domain of interest R for a logistic regression model. It can be seen that for different data sizes ($N = 100$ and 1000), the corresponding domains of interest are adjusted automatically to capture the high density regions of the posterior distribution.

When the high density region is irregular and can not be represented well by a rectangular box, this might not be an efficient approach. Later, we will discuss a more general approach for such cases.

4. Experiments. In this section, we compare our proposed method to standard HMC using several experiments in terms of sampling efficiency. We define sampling efficiency as time-normalized effective sample size (ESS). Given B MCMC samples for each parameter, we calculate the corresponding ESS $= B[1 + 2\Sigma_{k=1}^{K}\gamma(k)]^{-1}$, where $\Sigma_{k=1}^{K}\gamma(k)$ is the sum of K monotone sample autocorrelations [14]. We use the minimum ESS over all parameters normalized by the CPU time, s (in seconds), as the overall measure of efficiency: min(ESS)/s.

Empirical results show that both GHMC and Sparse Grid HMC (SGHMC) provide substantial improvement over standard HMC in terms of efficiency while maintaining relatively high acceptance rates.



HMC GHMC

FIG. 4.1. *HMC vs GHMC: logistic regression*

4.1. Logistic regression. For our first example, we sample $N = 100$ data points from a logistic regression model discussed in Section 3 and choose the domain of interest to be $[-3, 0.5] \times [-0.5, 3]$ and set the grid size to 0.1. Figure 4.1 shows posterior samples using standard HMC and GHMC. Note that they both converge to the target distribution and explore the parameter space quite well. Table 4.1 compares the performance of these algorithms based on 3200 MCMC iterations after burning the first 800 iterations. As we can see, GHMC outperforms standard HMC in terms of time-normalized ESS.

TABLE 4.1

Comparing HMC with GHMC using a logistic regression model. For each method, we provide the acceptance rate (AR), the CPU time (s) for each iteration and the time-normalized ESS

Method	AR	ESS(β_0, β_1)	s/Iteration	min ESS/s
HMC	0.9225	(3200, 3200)	7.0157E-4	1425.3707
GHMC	0.7981	(3200, 3200)	3.318E-4	3013.9031

4.2. Banana-shaped distribution. The potential energy function for the logistic regression model is quite similar to a Gaussian distribution model, where the resulting force function is relatively smooth. To investigate GHMC's ability to explore the parameter space with a more complicated geometry, we construct a banana-shaped posterior distribution of $\beta = (\beta_1, \beta_2|y)$ based on the following model:

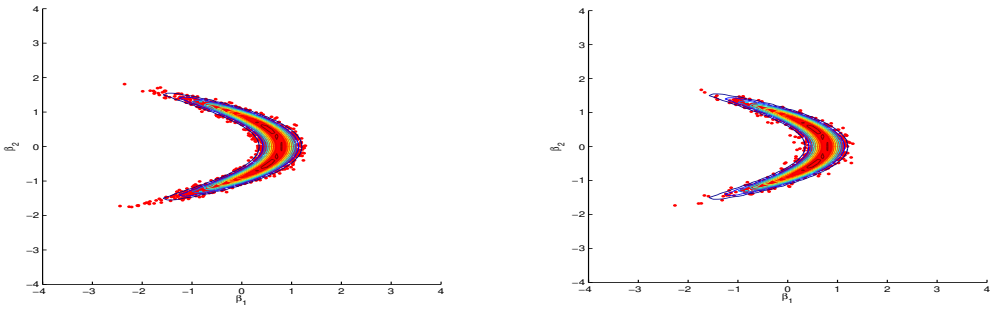
$$y|\beta \sim \mathcal{N}(\beta_1 + \beta_2^2, \sigma_y^2)$$
$$\beta \sim \mathcal{N}(0, \sigma_\beta^2)$$

The data $\{y_i\}_{i=1}^{100}$ are generated with $\beta_1 + \beta_2^2 = 1,\ \sigma_y = 2,\ \sigma_\beta = 1$. The potential energy function is

$$U(\beta) = \sum_{i=1}^{N} \frac{(y_i - \beta_1 - \beta_2^2)^2}{2\sigma_y^2} + \frac{\beta_1^2 + \beta_2^2}{2\sigma_\beta^2} \tag{4.1}$$

and the force function is

$$F(\beta) = -\frac{\partial U}{\partial \beta} = \frac{\sum_{i=1}^{N}(y_i - \beta_1 - \beta_2^2)}{\sigma_y^2} \cdot \begin{pmatrix} 1 \\ 2\beta_2 \end{pmatrix} - \frac{\beta}{\sigma_\beta^2} \tag{4.2}$$



HMC GHMC

FIG. 4.2. *HMC vs GHMC: banana-shaped distribution*

Here, we choose the domain of interest to be $[-4, 4] \times [-4, 4]$ and set grid size to 0.1. Figure 4.2 shows the samples for the posterior distribution using standard HMC and GHMC. As before, both methods converge to the target distribution and explore the parameter space quite well. Even though Banana-shaped distribution is more distorted and the force function is more complex, $\tilde{F}$ (grid size 0.1) still provide a good approximation to the true force function. Table 4.2 compares the performance of these algorithms based on 3200 MCMC iterations after burning the first 800 iterations. As before, GHMC outperforms standard HMC in terms of time-normalized ESS.

TABLE 4.2
Comparing HMC with GHMC using a banana-shaped distribution model. For each method, we provide the acceptance rate (AR), the CPU time (s) for each iteration and the time-normalized ESS

Method	AR	ESS(β_1, β_2)	s/Iteration	min ESS/s
HMC	0.9353	(2403, 1191.6)	3.8703E-4	962.1346
GHMC	0.6587	(893.8862, 766.2423)	1.4498E-4	1651.5917

4.3. Gaussian Process model. For our third example, we use a Gaussian process model. Posterior sampling for these models tends to be quite difficult due to the computation cost associated with inverting the covariance matrix. See Neal [21] and Rasmussen [23] for more details on Gaussian process. Here we construct a 2D Gaussian process with zero mean and the squared exponential covariance function,

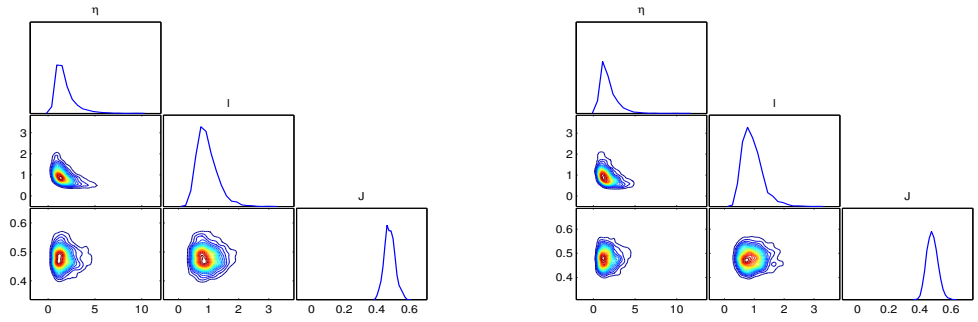
$$Y \sim \mathcal{N}(0, \Sigma), \quad \Sigma_{ij} = \eta \cdot \exp\left(-l\|x_i - x_j\|_2^2\right) + J \cdot \delta_{ij}$$

where η, l, J are positive hyperparameters with log-normal priors.

$$\log(\eta) \sim \mathcal{N}(-1, 1), \quad \log(l) \sim \mathcal{N}(-1, 1), \quad \log(J) \sim \mathcal{N}(-1, 1)$$

Let $\tilde{\eta} = \log(\eta),\ \tilde{l} = \log(l),\ \tilde{J} = \log(J)$, the potential energy function is

$$U(\tilde{\eta}, \tilde{l}, \tilde{J}) = \frac{1}{2}\log(|\Sigma|) + \frac{1}{2}Y^T\Sigma^{-1}Y + \frac{1}{2}\left[(\tilde{\eta}+1)^2 + (\tilde{l}+1)^2 + (\tilde{J}+1)^2\right]$$



HMC SGHMC

FIG. 4.3. *HMC vs SGHMC: Gaussian Process*

and the force function is

$$F(\beta) = -\frac{\partial U}{\partial \beta} = \frac{1}{2}\mathrm{tr}\left(\Sigma^{-1}\frac{\partial \Sigma}{\partial \beta}\right) - \frac{1}{2}Y^T\Sigma^{-1}\frac{\partial \Sigma}{\partial \beta}\Sigma^{-1}Y + \beta + 1, \quad \beta = (\tilde{\eta}, \tilde{l}, \tilde{J})^T$$

The domain of interest for $\beta = (\tilde{\eta}, \tilde{l}, \tilde{J})^T$ is set to be $[-1.6, 1.6]\times[-1.6, 1.6]\times[-1.2, 0.4]$ where we train a sparse gird interpolator to replace the force function. Figure 4.3 shows the samples from the posterior distribution given by standard HMC and SGHMC. Table. 4.3 compares the performance of the two algorithms based on 3200 MCMC iterations after 800 burn-in iterations. As we can see, SGHMC substantially outperform standard HMC.

TABLE 4.3

Comparing HMC with SGHMC using a Gaussian process model. For each method, we provide the acceptance rate (AR), the CPU time (s) for each iteration and the time-normalized ESS

Method	AR	ESS($\eta,\ l,\ J$)	s/Iteration	min ESS/s
HMC	0.9472	(1021.7, 1784.8, 3200)	2.3547E-1	1.3559
SGHMC	0.7066	(828.7, 1380.0, 3200)	2.9851E-2	8.6752

4.4. Elliptic PDE Inverse Problem. Our last example is a canonical inverse problem involving inference of the diffusion coefficient in an elliptic PDE ([12, 11]). The forward model is to solve a two dimensional elliptic PDE

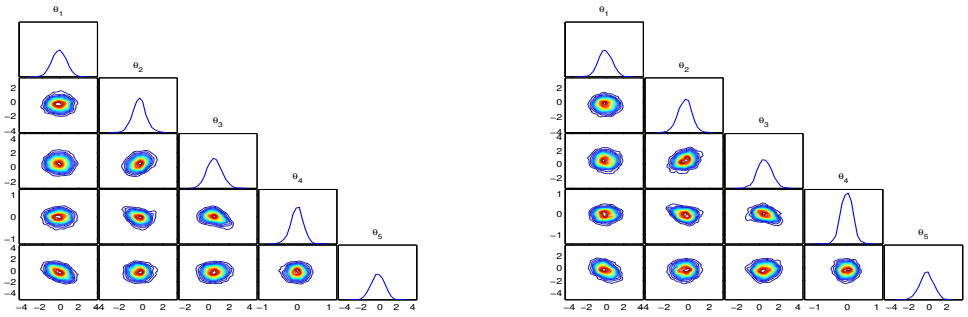
$$\nabla_{\boldsymbol{x}} \cdot (c(\boldsymbol{x}, \theta)\nabla_{\boldsymbol{x}} u(\boldsymbol{x}, \theta)) = 0 \tag{4.3}$$

where $\boldsymbol{x} = (x_1, x_2) \in [0, 1]^2$ is the spatial coordinate. The boundary conditions are

$$u(\boldsymbol{x}, \theta)|_{x_2=0} = x_1, \quad u(\boldsymbol{x}, \theta)|_{x_2=1} = 1 - x_1$$

$$\frac{\partial u(\boldsymbol{x}, \theta)}{\partial x_1}\Big|_{x_1=0} = 0, \quad \frac{\partial u(\boldsymbol{x}, \theta)}{\partial x_1}\Big|_{x_1=1} = 0$$

This PDE provides a simple model of steady-state flow in porous media. The coefficient c represents the permeability of a porous medium while u represents the pressure head. In this inverse problem,



HMC SGHMC

FIG. 4.4. *HMC vs SGHMC: an elliptic PDE inverse problem*

the objective of interest is to infer the unknown diffusion coefficient conditioned on observation data where Bayesian approach can be naturally adopted. A log-Gaussian process prior is given to the diffusivity field $c(\boldsymbol{x})$ with mean zero and an isotropic squared-exponential covariance kernel:

$$C(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sigma^2 \exp\left(-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2}{2l^2}\right)$$

for which we choose variance $\sigma^2 = 1$ and a length scale $l = 0.2$. With this prior, the field can be easily parameterized with a Karhunen-Loeve (K-L) expansion:

$$c(\boldsymbol{x}, \theta) \approx \exp\left(\sum_{i=1}^{d} \theta_i \sqrt{\lambda_i} v_i(\boldsymbol{x})\right)$$

where λ_i and $v_i(\boldsymbol{x})$ are the eigenvalues and eigenfunctions of the integral operator defined by the kernel C, and the parameter θ_i are endowed with independent standard normal priors, $\theta_i \sim \mathcal{N}(0, 1)$, which are the targets of inference. To reduce the dimension of this inference problem, the Karhunen-Loeve expansion is truncated at the first five modes ($d = 5$) and the corresponding mode weights $(\theta_1, \ldots, \theta_5)$ are conditioned on data. Data are generated by combining observations of the solution field (solve the PDE on a finer 51-by-51 grid) on a uniform 11×11 grid covering the unit square with additive independent Gaussian noise.

$$y_j = u(\boldsymbol{x}_j, \theta) + \epsilon_j, \quad \epsilon_j \sim \mathcal{N}(0, 0.1^2)$$

Consistent with results from previous examples, SGHMC performs substantially better than HMC (Table 4.4).

5. Approximate Target Distribution. So far, our proposed method has been based on using the exact target distribution and approximating the proposal generating mechanism only. We can improve the computation speed even more by using grid approximation for U in the correction step (accept/reject step) as well. In this case, the resulting sampler actually samples from an approximate distribution

$$Q(q) \propto \exp(-\tilde{U}(q))$$

TABLE 4.4
Comparing HMC with SGHMC using an elliptic PDE inverse problem. For each method, we provide the acceptance rate (AR), the CPU time (s) for each iteration and the time-normalized ESS

Method	AR	ESS	s/Iteration	min ESS/s
HMC	0.7719	$(991.8, 2091.2, 2831.0)$	$2.02E$-1	1.5343
SGHMC	0.6141	$(855.7, 1325.7, 1937.5)$	$6.1952E$-2	4.3165

instead of the target posterior distribution $P(q)$. The bound of the difference between these two distributions measured by the Kullback-Leibler divergence is shown in the following theorem.

THEOREM 5.1. *If U and V are energy functions corresponding to probability distributions P and Q*

$$P(q) \propto \exp(-U(q)), \quad Q(q) \propto \exp(-V(q))$$

then the Kullback-Leibler divergence between P and Q is bounded by

$$D_{KL}(P\|Q) \leq 2\|U - V\|_\infty$$

Proof.

$$\begin{aligned} D_{KL}(P\|Q) &= \int_{\mathbb{R}^D} P(q) \ln\left(\frac{P(q)}{Q(q)}\right)\, dq \\ &= \int_{\mathbb{R}^D} P(q)(V(q) - U(q))\, dq + \int_{\mathbb{R}^D} P(q) \ln\left(\frac{I_Q}{I_P}\right)\, dq \end{aligned}$$

where

$$I_P = \int_{\mathbb{R}^D} \exp(-U(q))\, dq, \quad I_Q = \int_{\mathbb{R}^D} \exp(-V(q))\, dq$$

since

$$\begin{aligned} I_Q &= \int_{\mathbb{R}^D} \exp(-V(q))\, dq = \int_{\mathbb{R}^D} \exp(-U(q)) \cdot \exp(-(V(q) - U(q)))\, dq \\ &\leq \exp(\|V - U\|_\infty) \cdot \int_{\mathbb{R}^D} \exp(-U(q))\, dq = \exp(\|V - U\|_\infty) \cdot I_P \end{aligned}$$

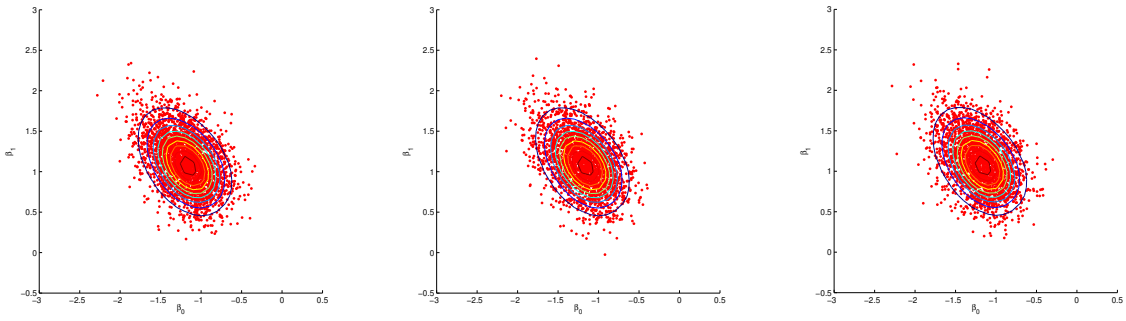
we have

$$D_{KL}(P\|Q) \leq \|V - U\|_\infty \cdot \int_{\mathbb{R}^D} P(q)\, dq + \|V - U\|_\infty \cdot \int_{\mathbb{R}^D} P(q)\, dq = 2\|V - U\|_\infty$$

□

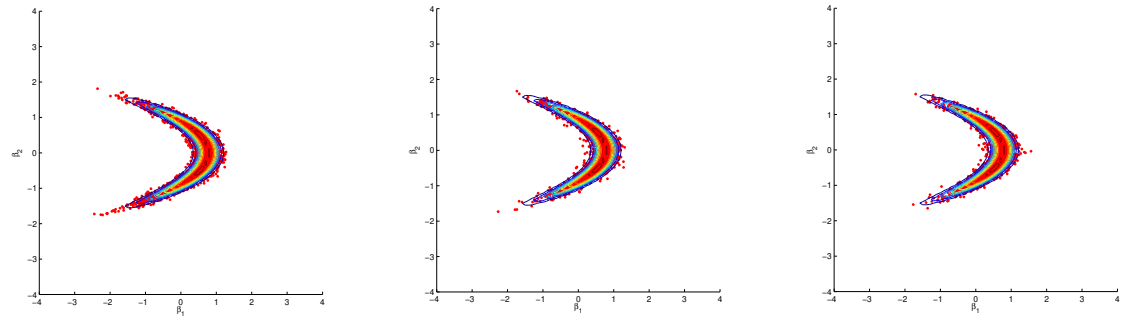
Note that if the potential energy function U is a smooth function, $\|\tilde{U} - U\|_\infty \to 0$ as the grid size goes to 0. By Theorem. 5.1, the resulting sampler will eventually converge to the target sampler.

We apply this method, called GHMC-complete, to the logistic regression and banana-shaped distribution examples discussed above. The results are shown in Figures 5.1 and 5.2. As we can see,



HMC GHMC GHMC-complete

FIG. 5.1. *HMC vs GHMC: logistic regression*



HMC GHMC GHMC-complete

FIG. 5.2. *HMC vs GHMC: banan-shaped distribution*

the posterior samples given by GHMC-complete in both cases match the exact samplers (HMC and GHMC) quite well. With appropriate grid size (around the step size), GHMC-complete can provide a high quality approximation to the standard HMC sampler. At the same time, computational efficiency has been substantially improved due to the fast computation of potential energy function in the correction step (Tables 5.1 and 5.2).

For the logistic regression example, Figure 5.3 shows the prediction accuracy vs. the run time for the three algorithms based on a test set. As we can see, the prediction accuracy (measured in terms of the average log-likelihood on the test data) of GHMC-complete increases faster compared to the other two methods. For computationally intensive models, the advantage of GHMC-complete will be more significant since the computation cost of the potential energy function becomes more expensive.

6. Discussion. Due to its ability of producing distant proposals with high acceptance probability, HMC can provide rapid exploration of the parameter space when sampling from the posterior distribution. However, the gradient computation to obtain essential geometric information prevents its application on computationally intensive problems when the data size is large. To address this issue, we have proposed a relaxed framework, where HMC can take advantage of the smoothness of

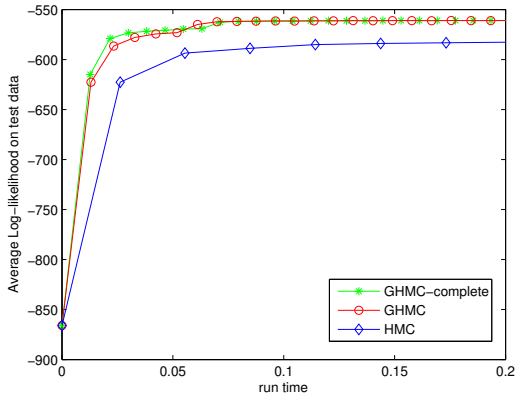


FIG. 5.3. *The average log-likelihood on test data for a logistic regression model*

TABLE 5.1

Comparing HMC with GHMC using a logistic regression model. For each method, we provide the acceptance rate (AR), the CPU time (s) for each iteration and the time-normalized ESS

Method	AR	ESS($\beta_0,\ \beta_1$)	s/Iteration	min ESS/s
HMC	0.9225	(3200, 3200)	7.0157E-4	1425.3707
GHMC	0.7981	(3200, 3200)	3.318E-4	3013.9031
GHMC-complete	0.7931	(3191.8, 3200)	2.9237E-4	3411.5275

TABLE 5.2

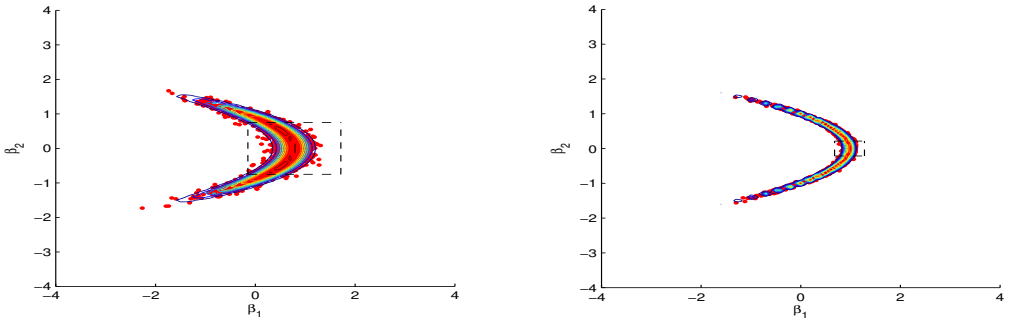
Comparing HMC with GHMC using a banana-shaped distribution model. For each method, we provide the acceptance rate (AR), the CPU time (s) for each iteration and the time-normalized ESS

Method	AR	ESS($\beta_1,\ \beta_2$)	s/Iteration	min ESS/s
HMC	0.9353	(2403, 1191.6)	3.8703E-4	962.1346
GHMC	0.6587	(893.8862, 766.2423)	1.4498E-4	1651.5917
GHMC-complete	0.6697	(980.1443, 796.2977)	1.2279E-4	2026.6108

the potential energy function U in parameter space to accelerate computation by using grid-based precomputing strategies. The key idea is to approximate the force field generated by the potential energy function U through interpolation of those precomputed field at grid points in each HMC iteration. Based on these ideas, two simple grid based algorithms, Naive Grid HMC and Sparse Grid HMC, are proposed and evaluated on several problems. Empirical results show that our approach can capture the main information needed for HMC's implementation at a lower computational cost. As a result, our method tends to be more effective than standard HMC.

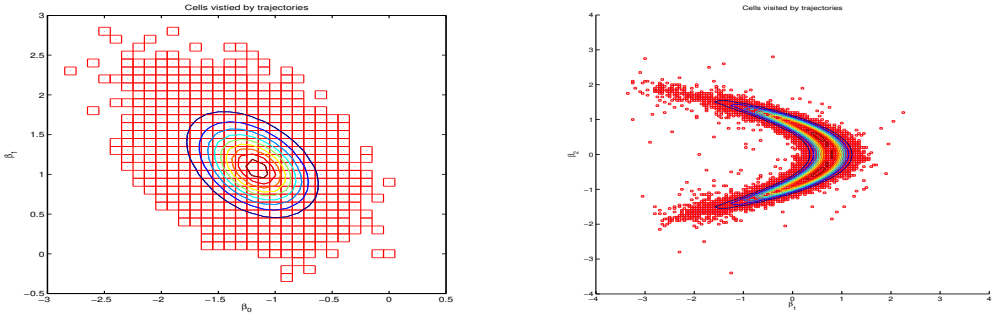
While quite effective in relatively low dimensional problems, extension of grid-based HMC to high dimensional problems could be quite challenging. Future research direction could involve finding effective strategies to alleviate this issue.

Another direction is to find more efficient method to locate the domain of interest. In subsection 3.5 we used Laplace's approximation for this purpose. As shown in Figure 5.4, this strategy might not be effective when the resulting Gaussian distribution is not a good approximation for the target distributions. An alternative and more general approach is based on following the trajectories of



(c) $N = 100$ (d) $N = 1000$

FIG. 5.4. *Domains of Interest via Laplace's approximation for the banana shaped distribution.*



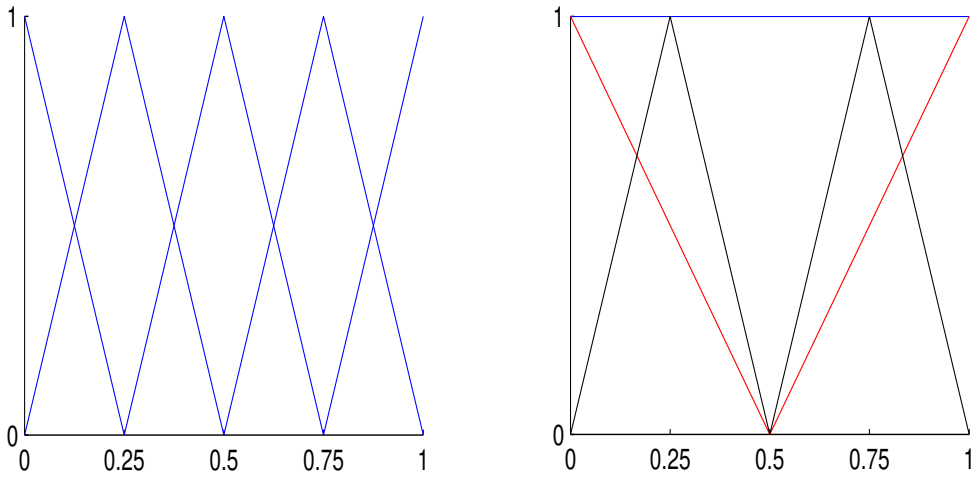
(a) logistic regression (b) banana shaped distribution

FIG. 5.5. *Domains of Interest via early trajectories.*

the burn-in samples. Even though MCMC samplers might not converge to the target distribution in the early stage, those trajectories can capture the high density region to some extent. Figure 5.5 shows the cells visited by those early trajectories for the logistic regression example and the banana shaped distribution example.

The proposed precomputing strategy is not limited to HMC only. In fact, it can be integrated with other MCMC methods involving expensive computation of redundant information. For example, Fisher information matrices can be precomputed at each cell center to accelerate Riemannian Manifold HMC [15].

Acknowledgments. This work was partially supported by NSF DMS- 1418422. We also appreciate the discussions with Shiwei Lan and Tian Chen.



(a) Nodal basis (b) Hierarchical basis

FIG. A.1. *Piecewise linear Nodal basis (a) and hierarchical functions (b) with support nodes* $x_j^i \in X_\Delta^i, i = 1, 2, 3$ *for the Clenshaw-Curtis grid*

Appendix. More On Sparse Grid.

A.1. Construction of sparse grid and basis functions. The Clenshaw-Curtis type sparse grid H^{CC} introduced in subsection 3.4.2 is constructed from the following formulas. Here, the x_j^i are defined as

$$x_j^i = \begin{cases} (j-1)/(m_i-1), & j = 1, \ldots, m_i, m_i > 1 \\ 0.5, & j = 1, m_i = 1 \end{cases}$$

In order to obtain nested sets of points, the number of nodes is given by

$$m_1 = 1 \quad \text{and} \quad m_i = 2^{i-1} + 1 \text{ for } i > 1$$

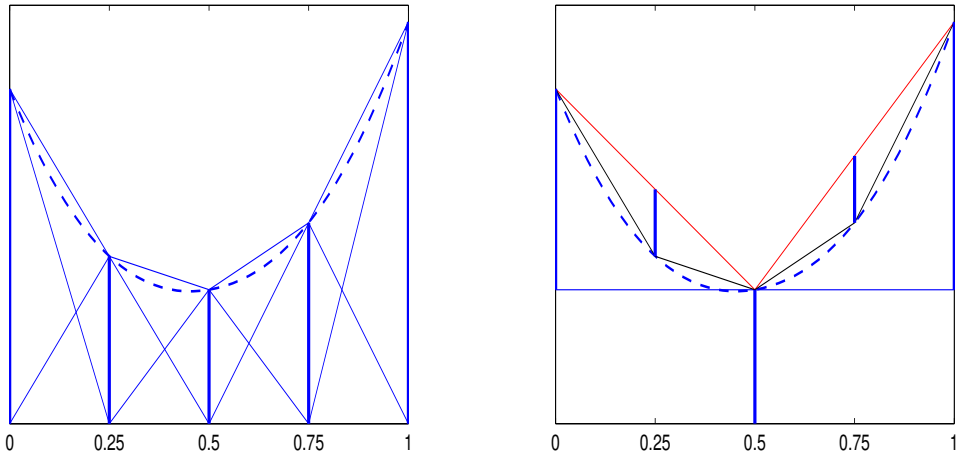
Piecewise linear basis functions a can be used for the univariate interpolation formulas $U^i(f)$.

$$a_1^1(x) = 1, \quad a_j^i(x) = \begin{cases} 1 - (m_i - 1) \cdot |x - x_j^i|, & |x - x_j^i| < 1/(m_i - 1), \\ 0, & \text{otherwise} \end{cases}$$

for $i > 1$ and $j = 1, \ldots, m_i$.

A.2. Derivation of the hierarchical form. With the selection of nested sets of points, we can easily transform the univariate nodal basis into the hierarchical one. By definition, we have

$$\begin{aligned} \Delta^i(f) &= U^i(f) - U^{i-1}(f) \\ &= \sum_{j=1}^{m_i} f(x_j^i) \cdot a_j^i - \sum_{j=1}^{m_i} U^{i-1}(f)(x_j^i) \cdot a_j^i \\ &= \sum_{j=1}^{m_i} \left(f(x_j^i) - U^{i-1}(f)(x_j^i) \right) \cdot a_j^i \end{aligned}$$



(a) Nodal basis (b) Hierarchical basis

FIG. A.2. *Piecewise linear interpolation: Nodal versus Hierarchical*

since $f(x_j^i) - U^{i-1}(f)(x_j^i) = 0,\ \forall\ x_j^i \in X^{i-1}$,

$$\Delta^i(f) = \sum_{x_j^i \in X_\Delta^i} \left(f(x_j^i) - U^{i-1}(f)(x_j^i)\right) \cdot a_j^i \tag{A.1}$$

From (A.1) we note that for all $\Delta^i(f)$, only the basis functions belonging to the grid points that have not yet occurred in a previous set $X^{i-k},\ 1 \le k \le i-1$ are involved. Fig.A.1 gives a comparison of the nodal and the hierarchical basis functions and Fig.A.2 shows the construction of the interpolation formula using nodal basis functions and function values versus using hierarchical basis functions and hierarchical surpluses for a univariate function f . Both figures are reproductions based on Klimke and Wohlmuth [18].

Applying the tensor product formula (3.7) with Δ^i given in (A.1), the hierarchical update in the Smolyak algorithm (3.8) now can be rewritten as

$$\begin{aligned}\Delta A_{q,d}(f) &= \sum_{|\boldsymbol{i}|=q} (\Delta^{i_1} \otimes \cdots \otimes \Delta^{i_d})(f) \\ &= \sum_{|\boldsymbol{i}|=q} \sum_{x_{j_1}^{i_1} \in X_\Delta^{i_1}} \cdots \sum_{x_{j_d}^{i_d} \in X_\Delta^{i_d}} \left(f(x_{j_1}^{i_1}, \ldots, x_{j_d}^{i_d}) - A_{q-1,d}(f)(x_{j_1}^{i_1}, \ldots, x_{j_d})\right) \cdot (a_{j_1}^{i_1} \otimes \cdots \otimes a_{j_d}^{i_d})\end{aligned}$$

REFERENCES

[1] Y. Ahmadian, J. W. Pillow, and L. Paninski, *Efficient Markov Chain Monte Carlo methods for decoding neural spike trains*, Neural Computation, 23 (2011), pp. 46–96.

[2] S. Ahn, Y. Chen, and M. Welling, *Distributed and adaptive darting monte carlo through regenerations*, International Conference on Artificial Intelligence and Statistics, (2013).

[3] S. Ahn, B. Shahbaba, and M. Welling, *Distributed Stochastic Gradient MCMC*, in International Conference on Machine Learning, 2014.

[4] B. J. Alder and T. E. Wainwright, *Studies in Molecular Dynamics. I. General Method*, The Journal of Chemical Physics, 31 (1959), pp. 459–466.

[5] V. Barthelmann, E. Novak, and K. Ritter, *High dimensional polynomial interpolation on sparse grids*, Adv. Comput. Math., 12 (2000), pp. 273–288.

[6] A. Beskos, F. J. Pinski, J. M. Sanz-Serna, and A. M. Stuart, *Hybrid Monte-Carlo on Hilbert spaces*, Stochastic Processes and their Applications, 121 (2011), pp. 2201–2230.

[7] H. J. Bungartz, *Finite Elements of Higher Order on Sparse Grids*, Sharker Verlag., 1998.

[8] H. J. Bungartz and M. Griebel, *Sparse grids*, Acta Numerica., 13 (2004), pp. 147–169.

[9] B. Calderhead and M. Sustik, *Sparse approximate manifolds for differential geometric mcmc*, in Advances in Neural Information Processing Systems 25, P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, eds., 2012, pp. 2888–2896.

[10] T. Chen, E. B. Fox, and C. Guestrin, *Stochastic gradient hamiltonian monte carlo*, Preprint, (2014).

[11] P. R. Conard, Y. M. Marzouk, N. S. Pillai, and A. Smith, *Asymptotically exact mcmc algorithms via local approximations of computationally intensive models*, Arxiv preprint arXiv:1402.1694v1, (2014), pp. 1–38.

[12] M. Dashti and A. M. Stuart, *Uncertainty quantification and weak approximation of an elliptic inverse problem*, SIAM Journal of Numerical Analysis., 6 (2011), pp. 2524–2542.

[13] S. Duane, A. D. Kennedy, B J. Pendleton, and D. Roweth, *Hybrid Monte Carlo*, Physics Letters B, 195 (1987), pp. 216 – 222.

[14] C. J. Geyer, *Practical Markov Chain Monte Carlo*, Statistical Science, 7 (1992), pp. 473–483.

[15] M. Girolami and B. Calderhead, *Riemann manifold Langevin and Hamiltonian Monte Carlo methods*, Journal of the Royal Statistical Society, Series B, (with discussion) 73 (2011), pp. 123–214.

[16] M. Hoffman and A. Gelman, *The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo.* arxiv.org/abs/1111.4246, 2011.

[17] Matthew D. Hoffman, David M. Blei, and Francis R. Bach, *Online learning for latent dirichlet allocation.*, in NIPS, John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta, eds., Curran Associates, Inc., 2010, pp. 856–864.

[18] A. Klimke and B. Wohlmuth, *Algorithm 847: spinterp: Piecewise multilinear hierarchical sparse grid interpolation in matlab*, 31 (2005), pp. 561–579.

[19] S. Lan, B. Zhou, and B. Shahbaba, *Spherical Hamiltonian Monte Carlo for Constrained Target Distributions*, in 31th International Conference on Machine Learning (to appear), 2014.

[20] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, *Equation of State Calculations by Fast Computing Machines*, The Journal of Chemical Physics, 21 (1953), pp. 1087–1092.

[21] R. M. Neal, *Regression and classification using Gaussian process priors*, Bayesian Statistics, 6 (1998), pp. 471–501.

[22] ———, *MCMC using Hamiltonian dynamics*, in Handbook of Markov Chain Monte Carlo, S. Brooks, A. Gelman, G. Jones, and X. L. Meng, eds., Chapman and Hall/CRC, 2011, pp. 113–162.

[23] C. E. Rasmussen, *Evaluation of Gaussian Processes and other Methods for Non-Linear Regression*, PhD Thesis, Department of Computer Science, University of Toronto., 1996.

[24] ———, *Gaussian processes to speed up hybrid monte carlo for expensive bayesian integrals*, Bayesian Statistics, (2003), pp. 651–659.

[25] B. Shahbaba, S. Lan, W.O. Johnson, and R.M. Neal, *Split Hamiltonian Monte Carlo*, Statistics and Computing, 24 (2014), pp. 339–349.

[26] G. W. Wasilkowski and H. Wozoniakowski, *Explicit cost bounds of algorithms for multivariate tensor product problems*, J. Complexity., 11 (1995), pp. 1–56.

[27] M. Welling and Y.W. Teh, *Bayesian learning via stochastic gradient langevin dynamics*, in Proceedings of the 28th International Conference on Machine Learning (ICML), 2011, pp. 681–688.