Version of Record: https://www.sciencedirect.com/science/article/pii/S0021999121000425
Manuscript_507b836edc7e64d0bdd54bffcd913705

An adaptive Hessian approximated stochastic gradient MCMC method

Yating Wang*[1], Wei Deng†[1], and Guang Lin‡[2]

[1]Department of Mathematics, Purdue University, West Lafayette, IN 47907, USA

[2]Department of Mathematics, School of Mechanical Engineering, Department of Statistics (Courtesy), Department of Earth, Atmospheric, and Planetary Sciences (Courtesy), Purdue University, West Lafayette, IN 47907, USA

January 15, 2021

Abstract

Bayesian approaches have been successfully integrated into training deep neural networks. One popular family is stochastic gradient Markov chain Monte Carlo methods (SG-MCMC), which have gained increasing interest due to their ability to handle large datasets and the potential to avoid overfitting. Although standard SG-MCMC methods have shown great performance in a variety of problems, they may be inefficient when the random variables in the target posterior densities have scale differences or are highly correlated. In this work, we present an adaptive Hessian approximated stochastic gradient MCMC method to incorporate local geometric information while sampling from the posterior. The idea is to apply stochastic approximation (SA) to sequentially update a preconditioning matrix at each iteration. The preconditioner possesses second-order information and can guide the random walk of a sampler efficiently. Instead of computing and saving the full Hessian of the log posterior, we use limited memory of the samples and their stochastic gradients to approximate the inverse Hessian-vector multiplication in the updating formula. Moreover, by smoothly optimizing the preconditioning matrix via SA, our proposed algorithm can asymptotically converge to the target distribution with a controllable bias under mild conditions. To reduce the training and testing computational burden, we adopt a magnitude-based weight pruning method to enforce the sparsity of the network. Our method is user-friendly and demonstrates better learning results compared to standard SG-MCMC

*E-mail: wang4190@purdue.edu
†E-mail: deng106@purdue.edu
‡Corresponding Author, Fax: 765 494 0548; Tel: 765 494 1965; E-mail: guanglin@purdue.edu

updating rules. The approximation of inverse Hessian alleviates storage and computational complexities for large dimensional models. Numerical experiments are performed on several problems, including sampling from 2D correlated distribution, synthetic regression problems, and learning the numerical solutions of heterogeneous elliptic PDE. The numerical results demonstrate great improvement in both the convergence rate and accuracy.

Keywords— Adaptive Bayesian method, deep learning, Hessian approximated stochastic gradient MCMC, stochastic approximation, limited memory BFGS, highly correlated density

1 Introduction

Deep learning has gained increasing interest in many areas due to its performance when dealing with large scale datasets. One important aspect of their successes in handling large datasets is that they process a small batch of data at each iteration to estimate the gradient of a cost function and update model parameters using gradient descent with a small step size. Bayesian approaches consider uncertainty in model parameters and help to improve the robustness in model learning. MCMC, as one of the most fashionable methods in Bayesian learning, is known for its asymptotic properties. However, it usually requires computations using the whole dataset, which is not feasible in large scale learning.

In recent years, many efforts have been made to bring Bayesian methods into the learning of DNNs [1, 24, 12]. One of the most popular approaches is stochastic gradient Langevin dynamics (SGLD) [24]. It is a stochastic gradient MCMC algorithm that originates from the discretization of Langevin diffusion. Similar to stochastic gradient descent (SGD), SGLD using mini-batches to approximate the gradients in the loss function. However, it injects a suitable amount of noise when updating parameters so that the sample variance matches the posterior variance. Moreover, with decreasing step sizes, it avoids the Metropolis-Hastings accept-reject step during sampling. It joins the stochastic optimization algorithm which resembles SGD, with Langevin dynamics which injects noise in the parameter updating formula. By injecting the right amount of noise, the method ensures that the trajectory of parameters will converge to the true posterior, rather than the MAP [24, 21, 5].

However, due to the complexity of DNN architecture, the model parameters may have complicated posterior density functions [10, 14, 6]. When the parameters have different scales in different directions, it may be inefficient if adopting a common step size. It becomes even more sophisticated if the target densities are highly correlated. There have been a lot of methods in the optimization community to overcome these difficulties and accelerate the gradient descent, such as preconditioning and stochastic Newton-type method [9, 26, 4, 3]. However, directly applying these methods to SGLD will not produce a correct MCMC scheme [14, 20, 16] in general. As indicated in [25, 13, 16], from another point of view, one can directly consider a Langevin diffusion on a Riemann manifold which described the geometric structure for the probability model. To ensure the diffusion has an invariant density, one needs to choose drift and volatility according to the Fokker–Planck equation, thus resulting in an additional drift term Γ. Several attempts have been made starting from the discretization of Riemann Langevin dynamics, to incorporate the underlying geometry according to the metric tensor in the sampling algorithm such that constant step size is adequate along with all directions. These methods also replace the gradient of a cost function using estimation from mini-batches as in SGLD. For example, stochastic gradient Riemann Langevin dynamics (SGRLD) [18] incorporates local curvature information by adopting the expected Fisher information as its

metric tensor. However, the full second-order Fisher information is intractable to obtain in many applications.

Preconditioned SGLD (PSGLD) is a computationally efficient method where a diagonal preconditioning matrix is employed as the metric tensor. In [14], the authors adopt a diagonal preconditioner where it is updated sequentially taking into account the current gradient and preconditioning matrix in the previous time step. This type of preconditioner can handle scale differences in the target density but may not be sufficient for highly correlated densities. Moreover, the correction term Γ needs the computation of third-order derivatives, and ignoring the term in the updating equation will introduce a permanent bias on the MSE [14]. To tackle these issues, a Hessian approximated stochastic gradient MCMC method (HAMCMC) [20] is studied, and it uses the local Hessian of the negative log posterior as an approximation to the full expected Fisher information. Instead of computing and storing the Hessian matrix, the limited memory BFGS (L-BFGS) algorithm [15, 4] is employed to approximate the product of inverse Hessian and gradient vectors. The idea is to reduce the computation and storage burden while maintaining accuracy. In addition, the current parameter at time step t is updated based on the sample at the previous time step $t-M$, and the approximated Hessian is computed using a history of samples at time steps $\{t-2M+1,\cdots,t-M+1,t-M-1,\cdots,t-1\}$. They claim that the correction term Γ vanishes due to this construction. However, when M is large, there will be a large gap between the two samples in the updating formula. Additionally, the method requires a relatively larger memory size $2M-2$ compared to standard memory size M.

In this paper, we propose a stochastic Hessian approximated MCMC algorithm with the help of stochastic approximation (SA) [2] to adaptively approximate the preconditioning matrix which involves the Hessian information. SA methods are typically used for root-finding problems or optimization problems in an iterative manner. It was first developed by Robbins and Monro [19], and serves as a typical framework in adaptive algorithms and control of stochastic systems. Let $H(\boldsymbol{\beta},\boldsymbol{\theta})$ be a random output function. To find the root of the mean field function $h(\boldsymbol{\theta})=\int H(\boldsymbol{\beta},\boldsymbol{\theta})\pi(\boldsymbol{\beta})d\boldsymbol{\beta}$, instead of evaluating $h(\boldsymbol{\theta})$ directly, SA adopts an adaptive method which resembles an online forms expectation–maximization approach. In each iteration, it samples $\boldsymbol{\beta_{k+1}}$ from a transition kernel which has the invariant distribution $\pi(\boldsymbol{\beta})$, and update the latent parameters $\boldsymbol{\theta}$ via the equation of the form $\boldsymbol{\theta_{k+1}}=\boldsymbol{\theta_k}+\omega_{k+1}H(\boldsymbol{\beta_{k+1}},\boldsymbol{\theta_k})$, with a suitable choice of sequences $\{\omega_k\}$ as the step size. Through this iterative process, $\boldsymbol{\theta}_k$ will converge in L_2 to the root of the target function $h(\boldsymbol{\theta})$ under mild conditions. The SA approach naturally fits in our training of a Bayesian model and sequentially updates preconditioning matrices. Moreover, we apply an iterative pruning strategy during the training to ensure the sparsity of the neural network. For simplicity, we use a magnitude-based criterion to prune weights in some densely connected layers given a user-defined sparse rate. The pruning technique not only behaves in a manner of greedy algorithms to remove redundant connections in the over-parameterized network but also allows the rebirth of pruned important connections to ensure robustness. Compared with HAMCMC, our proposed method (HAMCMC-SA) requires fewer samples in the L-BFGS algorithm. We prove that the samples generated from the proposed algorithm weakly converge to the true posterior with a controllable bias introduced by stochastic approximation. The advantages of our proposed algorithm are (1) user-friendly: the implementation is more straightforward, the parameter at time step t is updated based on the sample at the previous time step $t-1$ and there is no gap in the updating formula, (2) efficient: it requires less computation and memories, which is important in applications which require to run a very large-scale computational model, (3) the bias introduced by the algorithm is controllable and can be analyzed theoretically. Moreover, we adopt a magnitude-based weight pruning method to enforce the sparsity of the network, which further reduces the training and testing computational

cost.

The plan of the paper is as follows. In Section 2, we review some backgrounds in Langevin dynamics, Riemann Langevin dynamics, and some stochastic gradient MCMC algorithms. In Section 3, our main algorithm is proposed. We first present a detailed online damped L-BFGS algorithm which is used to approximate the inverse Hessian-vector product and discuss the properties of the approximated inverse Hessian. Next, the adaptive Hessian approximated MCMC algorithm with the stochastic approximation to the preconditioning matrix is presented. Its convergence is discussed in Section 4. Applying the proposed method to a simple 2D Gaussian distribution, a large-p-small-n regression problem, and to solve elliptic problems with varying source terms or heterogeneous coefficients, we demonstrate the numerical examples in Section 5 and conclude in Section 6.

2 Preliminary

First, we present backgrounds on SGLD, preconditioned SGLD, and Hessian approximated SGLD.

2.1 Langevin Dynamics and SGLD

Denote by $\boldsymbol{\beta}$ the model parameters in DNN. Let $D = \{d_i\}_{i=1}^N$ be the training dataset, where $d_i = (x_i, y_i)$ is an input-output pair. Let $p(\boldsymbol{\beta})$ be a prior distribution, and $p(d|\boldsymbol{\beta})$ be the likelihood function. The posterior distribution is then $p(\boldsymbol{\beta}|D) \propto p(\boldsymbol{\beta}) \prod_{i=1}^N p(d_i|\boldsymbol{\beta})$. The stochastic differential equation (SDE) which yields an invariant distribution $p(\boldsymbol{\beta}|D)$

$$d\boldsymbol{\beta}(t) = \nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}(t))dt + \sqrt{2}dW_t \tag{1}$$

where W_t is a Brownian motion and

$$\nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}) = \nabla_{\boldsymbol{\beta}} \log p(\boldsymbol{\beta}) + \sum_{i=1}^N \nabla_{\boldsymbol{\beta}} \log p(d_i|\boldsymbol{\beta})$$

The likelihood for regression problem can be rewritten as

$$p(d_k|\boldsymbol{\beta}, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\{ - \frac{\sum\limits_{x_i^k \in d_k} (y_i^k - \mathcal{F}(x_i^k; \boldsymbol{\beta}))^2}{2\sigma^2} \}$$

where $\mathcal{F}$ denotes a model describing the input-output map between x_i^k and y_i^k.

SGLD is a posterior Bayesian sampling method originates from the discretization of the SDE (1) and combines the idea from stochastic gradient algorithms. The loss gradient can be approximated efficiently using mini-batches, and the uncertainty in the model parameter can be captured in Bayesian learning. The model parameters update as follows:

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \epsilon_k \nabla_{\boldsymbol{\beta}} \tilde{L}(\boldsymbol{\beta}_k) + \mathcal{N}(0, 2\epsilon_k \tau^{-1})$$

where ϵ_k is the learning rate and

$$\nabla_{\boldsymbol{\beta}} \tilde{L}(\boldsymbol{\beta}_k) = \nabla_{\boldsymbol{\beta}} \log p(\boldsymbol{\beta}) + \frac{N}{n} \sum_{i=1}^n \nabla_{\boldsymbol{\beta}} \log p(d_{ki}|\boldsymbol{\beta})$$

is the stochastic gradient computed from a mini-batch $d_k = \{d_{k1}, \cdots, d_{kn}\}$.

2.2 Reimann Langevin Dynamics and HAMCMC

Stochastic Gradient Riemann Langevin Dynamics (SGRLD) [18] is a generalization of SGLD on a Riemannian manifold. If the components of the model parameter $\boldsymbol{\beta}$ possess different scales or are highly correlated, the invariant probability distribution for the Langevin equation is not isotropic, using standard Euclidian distance may lead to slow mixing. Given some metric tensor $G^{-1}(\boldsymbol{\beta})$, the SDE defining the Langevin diffusion with stationary distribution $p(\boldsymbol{\beta}|D)$ on a Riemann manifold is

$$d\boldsymbol{\beta}(t) = \left[G(\boldsymbol{\beta}(t))\nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}(t)) + \Gamma(\boldsymbol{\beta}(t))\right] dt + \sqrt{2G(\boldsymbol{\beta}(t))} dW_t \tag{2}$$

where $\Gamma_i(\boldsymbol{\beta}) = \sum_j \frac{\partial G_{ij}(\boldsymbol{\beta})}{\partial \beta_j}$. We note that $\Gamma(\boldsymbol{\beta})$ corresponds to variations in local curvature on the manifold and is equal to zero for a constant curvature. It is shown that, the invariant distribution of the dynamics (2) is $p(\beta|\mathcal{D}) \propto \exp L(\boldsymbol{\beta})$, and it is unique if $G^{-1}(\boldsymbol{\beta})$ is positive definite [16].

In this case, the parameter updates can be guided using the geometric information of this manifold:

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \epsilon_k \left[G(\boldsymbol{\beta}_k)\nabla_{\boldsymbol{\beta}} \tilde{L}(\boldsymbol{\beta}_k) + \Gamma(\boldsymbol{\beta}_k)\right] + \sqrt{2\epsilon_k \tau^{-1} G(\boldsymbol{\beta}_k)} z_k \tag{3}$$

where $z_k \sim \mathcal{N}(0, I)$.

A natural choice for metric tensor is the expected Fisher information matrix, however, it is intractable in many cases. In [14], the authors introduce a diagonal preconditioner, which resembles the preconditioning matrix in RMsProp, to reduce computational cost. However, it is effective to handle the case when there are scale differences among model parameters, but may not be sufficient to deal with strongly correlated target densities. A Hessian-approximated MCMC [20] method (HAMCMC) was proposed to overcome this issue. The idea is to compute the local curvature of the target density by approximating local Hessian information via quasi-newton approaches. In particular, HAMCMC generates samples $\boldsymbol{\beta}_k$ based on $\boldsymbol{\beta}_{k-M}$, where $M \geq 2$, and uses a history of samples $\{\boldsymbol{\beta}_{k-2M+1}, \cdots, \boldsymbol{\beta}_{k-M-1}, \boldsymbol{\beta}_{k-M+1}, \cdots, \boldsymbol{\beta}_{k-1}\}$ to approximate inverse Hessian information via limited BFGS. By this construction, the authors claim that the approximated Hessian is independent of the base-line sample $\boldsymbol{\beta}_{k-M}$, thus the correction term $\Gamma(\boldsymbol{\beta}_k)$ can be ignored without introducing additional bias. However, if the memory size M is large, there will be a large gap between two neighboring samples in the update rule. This may require a larger regularizer to ensure positive definite L-BFGS approximations, which result in a preconditioning matrix close to the identity matrix.

In this work, we adopt the stochastic approximation (SA) idea to iteratively update the approximated inverse Hessian. In each step, we sample $\boldsymbol{\beta}_k$ based on $\boldsymbol{\beta}_{k-1}$, and approximate $G(\boldsymbol{\beta}_k)$ using history samples $\{\boldsymbol{\beta}_{k-M+1}, \cdots, \boldsymbol{\beta}_{k-1}\}$. Compared with HAMCMC, our proposed method (HAMCMC-SA) requires fewer samples in the memory.

3 Main Method

3.1 The online damped L-BFGS algorithm

Now, we describe the online damped L-BFGS algorithm to approximate the local inverse Hessian at each iteration. In this approach, the approximated inverse Hessian matrix does not need to be computed or stored explicitly, but an approximation to the matrix-vector product is updated using successive gradient vectors instead.

Suppose we have a history of samples $\{\boldsymbol{\beta}_{k-M+1}, \cdots, \boldsymbol{\beta}_{k-1}\}$, where M is the memory size. Let $s_k = \boldsymbol{\beta}_{k+1} - \boldsymbol{\beta}_k$ be the increment in samples, and $y_k = \nabla_{\boldsymbol{\beta}} \tilde{L}(\boldsymbol{\beta}_{k+1}, d_k) - \nabla_{\boldsymbol{\beta}} \tilde{L}(\boldsymbol{\beta}_k, d_k)$ be the

differences between sample gradients. We remark that, although it may seem more natural to compute $y_k = \nabla_\beta \tilde{L}(\boldsymbol{\beta}_{k+1}, d_{k+1}) - \nabla_\beta \tilde{L}(\boldsymbol{\beta}_k, d_k)$ for the stochastic gradient variation, but this may not guarantee convergence [17]. Here the stochastic gradients $\nabla_\beta \tilde{L}(\boldsymbol{\beta}_{k+1}, d_k)$ and $\nabla_\beta \tilde{L}(\boldsymbol{\beta}_k, d_k)$ are evaluated with respect to the same set of samples d_k, which refers to the online L-BFGS [17]. This procedure helps to avoid additional differences between noisy gradient estimates and will only be applied for determining the stochastic gradient variation in the approximation of inverse Hessian.

Another thing to mention is that, given an initial guess of Hessian approximation which is positive definite. If the gradients are not stochastic, one usually use y_k directly in the approximation of Hessian/inverse Hessian matrices. In this case, the curvature condition $s_k^t y_k > 0$ can be satisfied via a line search and it is essential to ensure the positiveness of Hessian approximation after p recursion steps. In our case, due to the stochasticity, the line search is not feasible. We apply a stochastic damped BFGS method [22], modify y_k to $\bar{y}_k$ and justify $s_k^t \bar{y}_k > 0$. From there we can see the Hessian approximation is positive definite (details are shown in Lemma 1).

$$\bar{y}_k = \theta_k y_k + (1 - \theta_k) s_k \tag{4}$$

where

$$\theta_k = \begin{cases} \dfrac{(1-r) s_k^T s_k}{s_k^T B_{k,0} s_k - s_k^T B_{k,0} y_k}, & \text{if } s_k^T y_k < r s_k^T B_{k,0} s_k \\ 1, & \text{otherwise} \end{cases}$$

where $0 < r < 1$ is a constant, $B_{k,0}$ is the initial guess of the Hessian at k-th step, $\theta_0 = 1$ at the initial step in the recursion process.

The approximation of Hessian employs the following updating formula:

$$B_{k,i+1} = B_{k,i} + \frac{\bar{y}_j \bar{y}_j^T}{\bar{y}_j^T s_j} - \frac{B_{k,i} s_j s_j^T B_{k,i}}{s_j^T B_{k,i} s_j} \tag{5}$$

where $j = k - M + i$, M denotes the memory size. The initial guess of the recursion is typically chosen to be $B_{k,0} = \gamma_k I$, where $\gamma_k = \max\{\frac{\bar{y}_k \bar{y}_k^T}{s_k \bar{y}_k^T}, \delta\}$ and $\delta > 0$ is a given constant which is usually set to be 1. Denote by B_k be the final approximation of the Hessian, and $\tilde{G}_k = B_k^{-1}$. After M recursions, we take $B_k = B_{k,M}$.

For the online damped L-BFGS approximation to the inverse Hessian, we have

$$\tilde{G}_{k,i+1} = (I - \frac{s_j \bar{y}_j^T}{\bar{y}_j^T s_j}) \tilde{G}_{k,i} (I - \frac{s_j \bar{y}_j^T}{\bar{y}_j^T s_j})^T + \frac{s_j s_j^T}{\bar{y}_j^T s_j} \tag{6}$$

The initial guess of the recursion is $\tilde{G}_{k,0} = \gamma_k^{-1} I$.

As for the $\sqrt{\tilde{G}_k} := S_k$ ($\tilde{G}_k = S_k S_k^T$), with initial guess $S_{k,0} = \sqrt{\gamma_k^{-1}} I$,

$$S_{k,i+1} = (I - p_j q_j^T) S_{k,i} \tag{7}$$

$$p_j = \frac{s_j}{s_j^T \bar{y}_j}, \qquad q_j = \sqrt{\frac{s_j^T \bar{y}_j}{s_j^T B_{k,i} s_j}} B_{k,i} s_j - \bar{y}_j \tag{8}$$

For brevity, we denote by $g_k = \nabla_\beta \tilde{L}(\boldsymbol{\beta})$ from now on. The online damped L-BFGS algorithm to compute $\tilde{G}_k g_k$ and $\sqrt{\tilde{G}_k} z_k$ use a two-loop recursion and is described in Algorithm 1. Using the two-loop iteration is a standard approach in the L-BFGS algorithm [15]. The idea is that one would

like to avoid saving/storing the entire Hessian matrix as in BFGS, but computing the matrix-vector product through the two-loop iteration and only save the resulting product vectors. It represents the (inverse) Hessian approximation implicitly and only has linear memory requirements. To be specific, the detailed computation for the term $\tilde{G}_k g_k$ is presented through line 1-line 4, line 10, and line 12-14 in Algorithm 1. After recursion, the output ξ is the final results for $\tilde{G}_k g_k$. Similarly, for $z_k \sim \mathcal{N}(0,1)$, $S_k z_k$ is computed through line 5-line 9, line 11, line 12 and line 15. The output η is the final results for $S_k z_k$ after recursion.

Algorithm 1 Online damped L-BFGS

INPUT: g_k, $z_k \sim \mathcal{N}(0,1)$, M, $B_{k,0} = \gamma_k I$, $S_{k,0} = 1/\sqrt{\gamma_k} I$, $\tilde{G}_{k,0} = \gamma_k^{-1} I$

OUTPUT: $\tilde{G}_{k,M} g_k = \xi, \quad S_{k,M} z_k = \eta$

1: $w \leftarrow g_k$

2: **for all** $i \leftarrow k : \min\{k - M + 1, 0\}$ **do**

3: $\quad \alpha_i \leftarrow \dfrac{s_i^T w}{\bar{y}_i^T s_i}$

4: $\quad w \leftarrow w - \alpha_i \bar{y}_i$

5: $a_1 \leftarrow B_{k,0} s_{k-M+1}, \quad T_{1,j} = B_{k,0} s_{k-M+j}, \quad j = 1, \cdots, M$

6: **for all** $i \leftarrow 2 : k$ **do**

7: $\quad$ **for all** $j \leftarrow i : k$ **do**

8: $\quad\quad T_{i,j} \leftarrow T_{i-1,j} + \dfrac{\bar{y}_{i-1} s_j^T}{s_{i-1}^T \bar{y}_{i-1}} y_{i-1} - \dfrac{a_{i-1} s_j}{s_{i-1}^T a_{i-1}} a_{i-1}$

9: $\quad a_i \leftarrow T_{i,i}$

10: $\xi \leftarrow \tilde{G}_{k,0} w$

11: $\eta \leftarrow S_{k,0} z_k$

12: **for all** $i \leftarrow \min\{k - M + 1, 0\} : k$ **do**

13: $\quad \beta_i \leftarrow \dfrac{\bar{y}_i^T \xi}{\bar{y}_i^T s_i}$

14: $\quad \xi \leftarrow \xi + (\alpha_i - \beta_i) s_i$

15: $\quad \eta \leftarrow \eta - \dfrac{a_i^T \eta}{\sqrt{s_i^T \bar{y}_i} \sqrt{a_i^T s_i}} s_i - \dfrac{\bar{y}_i^T \eta}{s_i^T \bar{y}_i} s_i$

Lemma 1. *Let $\bar{y}_j$ be defined in (4), if $B_{k,i}$ and $\tilde{G}_{k,i}$ are positive definite, then $B_{k,i+1}$ and $\tilde{G}_{k,i+1}$ generated by (5) and (6) are both positive definite.*

Proof. By (4), we can easily obtain

$$s_j^T \bar{y}_j = \begin{cases} r s_j^T B_{k,0} s_j, & \text{if } s_j^T y_j < r s_j^T B_{k,0} s_j \\ s_j^T y_j, & \text{otherwise} \end{cases}$$

Thus, $s_j^T \bar{y}_j \geq r s_j^T B_{k,0} s_j > 0$ since $B_{k,0}$ is positive definite. By positive definiteness of $\tilde{G}_{k,i}$, for any nonzero vector $x \in \mathbb{R}^d$, we have

$$\mathbf{x}^T \tilde{G}_{k,i} \mathbf{x} > 0$$

Then it's easy to see

$$\begin{aligned}\mathbf{x}^T\tilde{G}_{k,i+1}\mathbf{x} &= \mathbf{x}^T(I-\frac{s_j\bar{y}_j^T}{\bar{y}_j^Ts_j})\tilde{G}_{k,i}(I-\frac{s_j\bar{y}_j^T}{\bar{y}_j^Ts_j})^T\mathbf{x}+\frac{1}{\bar{y}_j^Ts_j}\mathbf{x}^Ts_js_j^T\mathbf{x}\\ &= \mathbf{z}^T\tilde{G}_{k,i}\mathbf{z}+\frac{1}{\bar{y}_j^Ts_j}(s_j^T\mathbf{x})^2>0\end{aligned}$$

where $\mathbf{z}=(I-\frac{s_j\bar{y}_j^T}{\bar{y}_j^Ts_j})^T\mathbf{x}$. Thus, $\tilde{G}_{k,i+1}$ is positive definite, so is $B_{k,i+1}$.

□

In the following, the notation $P \preceq Q$ means $Q-P$ is positive semidefinite for two matrices P and Q.

Assumption 1. *There exists a constant $A<\infty$ such that the Hessian $\mathcal{H}(\boldsymbol{\beta})=\nabla^2\tilde{L}(\beta)$ satisfies $\mathcal{H}(\boldsymbol{\beta})\preceq A$ for any $\boldsymbol{\beta}$.*

Lemma 2. *The eigenvalues of Hessian approximation B_k generated from iteration (5) with $B_{k,0}=\gamma_kI$ are uniformly bounded,*

$$\tilde{a}I \preceq B_k \preceq \tilde{A}I$$

where $\tilde{A}=d(\delta+A)+\frac{M}{r}\left(\frac{A^2}{\delta}+(\delta+A)+2A\right)$ and $\tilde{a}=\frac{1}{\tilde{A}^{d-1}}\left(\frac{r}{\tilde{A}}\right)^M\delta^{d+M}$, and d is the size of the matrix, M is the number of recursions in online L-BFGS updates.

Proof. Take trace of the matrix in both hands of the equation (5), we have

$$\mathrm{tr}(B_{k,i+1})=\mathrm{tr}(B_{k,i})+\frac{1}{\bar{y}_j^Ts_j}\mathrm{tr}(\bar{y}_j\bar{y}_j^T)-\frac{1}{s_j^TB_{k,i}s_j}\mathrm{tr}(B_{k,i}s_js_j^TB_{k,i}) \tag{9}$$

By the properties of trace of a matrix, the above equation can be simplified as

$$\mathrm{tr}(B_{k,i+1})=\mathrm{tr}(B_{k,i})+\frac{\bar{y}_j^T\bar{y}_j}{\bar{y}_j^Ts_j}-\frac{||B_{k,i}s_j||^2}{s_j^TB_{k,i}s_j}\leq \mathrm{tr}(B_{k,i})+\frac{\bar{y}_j^T\bar{y}_j}{\bar{y}_j^Ts_j}$$

since $\frac{||B_{k,i}s_j||^2}{s_j^TB_{k,i}s_j}>0$ by the positive definiteness of $B_{k,i}$.

Now we derive a bound for $\frac{\bar{y}_j^T\bar{y}_j}{\bar{y}_j^Ts_j}$. Since $B_{k,0}=\gamma_kI$,

$$\frac{\bar{y}_j^T\bar{y}_j}{\bar{y}_j^Ts_j}=\frac{||\theta_jy_j+(1-\theta_j)B_{k,0}s_j||^2}{rs_j^TB_{k,0}s_j}=\frac{1}{r}\left(\frac{\theta_j^2||y_j||^2}{\gamma_k||s_j||^2}+(1-\theta_j)^2\gamma_k+\frac{2\theta_j(1-\theta_j)\gamma_ky_j^Ts_j}{||s_j||^2}\right) \tag{10}$$

Denote by $\bar{\mathcal{H}}=\int_0^1\mathcal{H}(\beta_k+\tau(\beta_{k+1}-\beta_k))d\tau$ the mean of Hessian in the segment $[\beta_k,\beta_{k+1}]$, $\bar{\mathcal{H}}\preceq AI$. Due to the fact

$$\frac{\partial\nabla\tilde{L}(\beta_k+\tau(\beta_{k+1}-\beta_k))}{\partial\tau}=(\beta_{k+1}-\beta_k)\mathcal{H}(\beta_k+\tau(\beta_{k+1}-\beta_k)), \tag{11}$$

we have

$$\int_0^1(\beta_{k+1}-\beta_k)\mathcal{H}(\beta_k+\tau(\beta_{k+1}-\beta_k))d\tau=\nabla\tilde{L}(\beta_{k+1})-\nabla\tilde{L}(\beta_k), \tag{12}$$

by integrating (11) over $[0,1]$ in both sides of the equation. One can see from (12) that $\bar{\mathcal{H}} s_k = y_k$. Thus, the first term and third term in (10) can be bounded as follows

$$\frac{\theta_j^2 ||y_j||^2}{\gamma_k ||s_j||^2} \leq \frac{||\bar{\mathcal{H}} s_j||^2}{\gamma_k ||s_j||^2} \leq \frac{A^2}{\delta}$$
$$\frac{2\theta_j(1-\theta_j) y_j^T s_j}{||s_j||^2} \leq 2\frac{s_j^T \bar{\mathcal{H}} s_j}{||s_j||^2} \leq 2A$$

since $0 < \theta_j < 1$, and $\delta \leq \gamma_k \leq \delta + A$. Plug these estimates in (10), we get

$$\frac{\bar{y}_j^T \bar{y}_j}{\bar{y}_j^T s_j} \leq \frac{1}{r}\left(\frac{A^2}{\delta} + (\delta + A) + 2A\right)$$

Then (9) can be bounded as

$$\begin{aligned}
\mathrm{tr}(B_{k,i+1}) &\leq \mathrm{tr}(B_{k,i}) + \frac{1}{r}\left(\frac{A^2}{\delta} + (\delta + A) + 2A\right) \\
&\leq \mathrm{tr}(B_{k,0}) + \frac{M}{r}\left(\frac{A^2}{\delta} + (\delta + A) + 2A\right) \\
&\leq d(\delta + A) + \frac{M}{r}\left(\frac{A^2}{\delta} + (\delta + A) + 2A\right)
\end{aligned}$$

where d is the size of matrix $B_{k,0}$, M is the number of recursions.

Since $B_{k,i+1}$ is positive definite, and $\mathrm{tr}(B_{k,i+1})$ is the sum of all eigenvalues of $B_{k,i+1}$, the largest eigenvalue $\mu_{\max}$ of $B_{k,i+1}$ satisfies

$$\mu_{\max} \leq d(\delta + A) + \frac{M}{r}\left(\frac{A^2}{\delta} + (\delta + A) + 2A\right) := \tilde{A}$$

Thus the largest eigenvalue of $B_{k,i+1}$ is no greater than $\tilde{A}$.

On the other hand,

$$\det(B_{k,i+1}) = \det(B_{k,i}) \det\left(I + \frac{B_{k,i}^{-1} \bar{y}_j \bar{y}_j^T}{\bar{y}_j^T s_j} - \frac{s_j (B_{k,i} s_j)^T}{s_j^T B_{k,i} s_j}\right) \tag{13}$$

The second term in the right hand side of (13) is equivalent to

$$\det\left(I + \frac{B_{k,i}^{-1} \bar{y}_j \bar{y}_j^T}{\bar{y}_j^T s_j} - \frac{s_j (B_{k,i} s_j)^T}{s_j^T B_{k,i} s_j}\right) = (1 + u_1^T u_2)(1 + u_3^T u_4) - (u_1^T u_4)(u_2^T u_3)$$

where $u_1 = -s_j$, $u_2 = \dfrac{B_{k,i} s_j}{s_j^T B_{k,i} s_j}$, $u_3 = B_{k,i}^{-1} \bar{y}_j$, $u_4 = \dfrac{\bar{y}_j}{\bar{y}_j^T s_j}$.

It is easy to check that $u_1^T u_2 = -1$, $u_1^T u_4 = -1$, $u_2^T u_3 = \frac{s_j^T \bar{y}_j}{s_j^T B_{k,i} s_j}$, thus (13) implies

$$\det(B_{k,i+1}) = \det(B_{k,i}) \frac{s_j^T \bar{y}_j}{s_j^T B_{k,i} s_j} \geq \det(B_{k,i}) \frac{r\gamma_k ||s_j||^2}{\tilde{A} ||s_j||^2} = \det(B_{k,i}) \frac{r\gamma_k}{\tilde{A}}$$

since $s_j^T \bar{y}_j \geq r s_j^T B_{k,0} s_j \geq r\gamma_k ||s_j||^2$ and $s_j^T B_{k,i} s_j \leq A ||s_j||^2$.

By induction and using the fact that $\det(B_{k,0}) = \gamma_k^d$, we have

$$\det(B_{k,i+1}) \geq \det(B_{k,0}) \left(\frac{r\gamma_k}{\tilde{A}}\right)^M \geq \left(\frac{r}{\tilde{A}}\right)^M \gamma_k^{d+M} \geq \left(\frac{r}{\tilde{A}}\right)^M \delta^{d+M}$$

Since any eigenvalue of $B_{k,i+1}$ is no greater than $\tilde{A}$, and $\det(B_{k,i+1})$ is equal to the product of all eigenvalues, we have that for any specific eigenvalue μ_j of $B_{k,i+1}$

$$\mu_j \geq \frac{1}{\tilde{A}^{d-1}} \left(\frac{r}{\tilde{A}}\right)^M \delta^{d+M} := \tilde{a}$$

Thus, we have

$$\tilde{a}I \preceq B_k \preceq \tilde{A}I.$$

Furthermore,

$$\frac{1}{\tilde{A}} I \preceq \tilde{G}_k \preceq \frac{1}{\tilde{a}} I.$$

□

3.2 Adaptive Hessian-approximated SG-MCMC with iterative pruning

The adaptive Hessian-approximated stochastic gradient MCMC with iterative pruning is a mixture of optimization and sample algorithm, where the model parameters are sampled from (3), and the preconditioning matrix $G(\boldsymbol{\beta})$ is optimized iteratively.

The idea is to obtain the optimal G_* based on the asymptotically correct distribution $\pi(\boldsymbol{\beta})$ through stochastic approximation. We aim to get an estimate G_* which solves the fixed point equation $\int g_G(\boldsymbol{\beta})\pi(\boldsymbol{\beta})d\boldsymbol{\beta} = G_*$, where $g_G(\cdot)$ denotes some mapping to derive the optimal G given current $\boldsymbol{\beta}$.

Define the random output $H(\boldsymbol{\beta}, G) = g_G(\boldsymbol{\beta}) - G$ and its mean field function $h(G) = \mathbb{E}[H(\boldsymbol{\beta}, G)]$. In our approach, we approximate $g_G(\boldsymbol{\beta})$ using the damped online L-BFGS as described in Algorithm 1. This will result in a bias $\delta(M, n, \epsilon_k)$ at each step which includes the error introduced by using stochastic gradients, and the error introduced by using a limited memory instead of full memory. Here M is the memory size, n is the number of samples in a mini-batch. That is, we use

$$\tilde{H}(\boldsymbol{\beta}, G) = H(\boldsymbol{\beta}, G) + \delta(M, n, \epsilon_k), \tag{14}$$

where we assume $\mathbb{E}||\delta(M, n, \epsilon_k)||^2 \leq C_0^2$.

After sampling $\boldsymbol{\beta}_{k+1}$ using (3) with approximated preconditioning matrix G_k, one can then update G_{k+1} from the following recursion:

$$G_{k+1} = G_k + \omega_{k+1}\tilde{H}(\boldsymbol{\beta}_{k+1}, G_k). \tag{15}$$

In summary, the adaptive empirical Bayesian algorithm samples $\boldsymbol{\beta}$ and optimize $G(\boldsymbol{\beta})$ as in Algorithm 2.

4 Convergence analysis

In this section, we will discuss the convergence of stochastic approximation and the proposed algorithm.

Algorithm 2 HAMCMC-SA

INPUT: Initialize $\boldsymbol{\beta}_1$, M, p, $G_1 = I$, τ

for all $k \leftarrow 1 : \#iterations$ **do**
  $g(\boldsymbol{\beta}_k) \leftarrow \nabla_\beta \tilde{L}(\cdot|d_k)$
  $z_k \sim \mathcal{N}(0, I)$
  $\tilde{G}_k g_k$, $\tilde{G}_k g_{k-1}$, $\tilde{S}_k z_k$, $\tilde{S}_k z_{k-1}$ from Algorithm 1
  $G_k g_k \leftarrow (1-\omega_k) G_{k-1} g_k + \omega_k \tilde{G}_k g_k$
  $S_k z_k \leftarrow (1-\omega_k) S_{k-1} z_k + \omega_k \tilde{S}_k z_k$
  $\xi_k \leftarrow G_k g_k / ||G_k g_k||$
  $\eta_k \leftarrow S_k z_k / ||S_k z_k||$
  $\boldsymbol{\beta}_{k+1} \leftarrow \boldsymbol{\beta}_k + \epsilon_k \xi_k + \sqrt{2\epsilon_k \tau^{-1}} \eta_k$
  if Pruning **then**
    Prune the bottom -p% weights with the lowest magnitude
    Increase the sparse rate

4.1 Convergence of stochastic approximation of preconditioning matrix

Denote by $\vec{G}$ vectorization of a matrix G, we first state the following stability lemma.

Lemma 3. *The mean field function $h(G)$ satisfies $\forall G \in \mathbb{R}^{d\times d}$, where d is the dimension of $\boldsymbol{\beta}$, $\langle h(\vec{G}), \vec{G} - \vec{G}_* \rangle \leq -||\vec{G} - \vec{G}_*||^2$, where $||\cdot||$ denotes l_2 norm. The mean field system $\frac{d\vec{G}}{dt} = h(\vec{G})$ is stable and G_* is the asymptotically stable equilibrium.*

Proof. Since $H(\boldsymbol{\beta}, G) = g_G(\boldsymbol{\beta}) - G$, the mean field function $h(G)$ is

$$h(G) = \int (g_G(\boldsymbol{\beta}) - G)\, \pi(\boldsymbol{\beta}) d\boldsymbol{\beta} = G_* - G$$

Then,

$$\langle h(\vec{G}), \vec{G} - \vec{G}_* \rangle = -||\vec{G} - \vec{G}_*||^2 \leq -||\vec{G} - \vec{G}_*||^2$$

Consider the positive definite Lyapunov function $V(\vec{G}) = \frac{1}{2}||\vec{G}_* - \vec{G}||^2$, it's easy to see that $\langle \nabla V, \frac{d\vec{G}}{dt} \rangle = \langle \vec{G} - \vec{G}_*, \vec{G}_* - \vec{G} \rangle = -||\vec{G} - \vec{G}_*||^2 < 0$, which completes the proof.

□

Assumption 2. *The step size $\{\omega_k\}$ satisfies*

$$\sum_{k=1}^{\infty} \omega_k = +\infty, \qquad \sum_{k=1}^{\infty} \omega_k^2 < +\infty$$

$$\liminf_{k\to\infty} 2\frac{\omega_k}{\omega_{k+1}} + \frac{\omega_{k+1} - \omega_k}{\omega_{k+1}^2} > 0$$

In practice, one can choose $\omega_k = c_1(k + c_2)^{-\alpha}$ for $\alpha \in (0, 1]$) and constants c_1, c_2.

Lemma 4. *There exists $Q > 0$, such that $\sup \mathbb{E}||G_k||^2 \leq Q^2$.*

Proof. From Lemma 2, we have

$$\frac{1}{\tilde{A}} \preceq \tilde{G}_k \preceq \frac{1}{\tilde{a}}$$

We will prove by induction. For $k=0$, $\mathbb{E}||G_0||^2 \leq \frac{1}{\tilde{a}} := Q$. Assume we have $\mathbb{E}||G_k||^2 \leq Q$, then

$$\begin{aligned}
\mathbb{E}||G_{k+1}||^2 &= \mathbb{E}||(1-\omega_k)G_k + \omega_k \tilde{G}_{k+1}||^2 \\
&\leq (1-\omega_k)^2 \mathbb{E}||G_k||^2 + 2(1-\omega_k)\omega_k \sqrt{\mathbb{E}||G_k||^2 \mathbb{E}||\tilde{G}_{k+1}||^2} + \omega_k^2 \mathbb{E}||\tilde{G}_{k+1}||^2 \\
&\leq (1-\omega_k)^2 Q^2 + 2(1-\omega_k)\omega_k \sqrt{Q^2 (\frac{1}{\tilde{a}})^2} + \omega_k^2 (\frac{1}{\tilde{a}})^2 \leq Q^2.
\end{aligned}$$

This completes the proof. □

Assumption 3. *For all $G \in \Theta$, there exists a function $\mu_G(\boldsymbol{\beta})$ that solves the Poisson equation $\mu_G(\boldsymbol{\beta}) - \Pi_G \mu_G(\boldsymbol{\beta}) = H(G, \boldsymbol{\beta}) - h(G)$. There exists a constant C such that*

$$\mathbb{E}||\Pi_G \mu_G(\boldsymbol{\beta})|| \leq C$$
$$\mathbb{E}||\Pi_G \mu_G(\boldsymbol{\beta}) - \Pi_{G'} \mu_{G'}(\boldsymbol{\beta})|| \leq C||G - G'||$$

Here $||\cdot||$ denote the Frobenius norm.

Lemma 5. *There exists a constant $Q_2 > 0$ such that*

$$||\tilde{H}(\boldsymbol{\beta}, G)||^2 \leq Q_2(1 + ||G_k - G_*||^2) \tag{16}$$

Proof.

$$||H(\boldsymbol{\beta}, G)||^2 \leq 2||g_G(\boldsymbol{\beta})||^2 + 2||G_k||^2 \leq 2(\frac{1}{\tilde{a}})^2 + 2||G_k||^2 \leq C_1(1 + ||G_k||^2) \leq \tilde{C}_1(1 + ||G_k - G_*||^2).$$

Then

$$\begin{aligned}
||\tilde{H}(\boldsymbol{\beta}, G)||^2 &= ||H(\boldsymbol{\beta}, G) + \delta(M, n, \epsilon_k)||^2 \leq 2||H(\boldsymbol{\beta}, G)||^2 + 2||\delta(M, n, \epsilon_k)||^2 \\
&\leq 2\tilde{C}_1(1 + ||G_k - G_*||^2) + 2C_0^2 \leq Q_2(1 + ||G_k - G_*||^2)
\end{aligned}$$

where $Q_2 = 2\tilde{C}_1 + 2C_0^2$. □

Lemma 6. *Let k_0 be an integer which satisfies*

$$\inf_{k \geq k_0} \frac{\omega_{k+1} - \omega_k}{\omega_k \omega_{k+1}} + 2 - Q\omega_{k+1} > 0$$

Then $\forall k \geq k_0$, the sequence $\{\Lambda_k^K\}_{k=k_0}^K$ is increasing, where

$$\Lambda_k^K = \begin{cases} 2\omega_k \prod_{j=k}^{K-1} (1 - 2\omega_{k+1} + Q\omega_{k+1}^2), & \text{if } k < K \\ 2\omega_k, & \text{if } k \geq K \end{cases}$$

Lemma 7. *There exists λ_0 and k_0 such that $\forall \lambda \geq \lambda_0$ and $\forall k \geq k_0$, the sequence $\{\psi_k\}_{k=1}^\infty$ with $\psi_k = \lambda\omega_k + 2Q \sup_{i \geq k_0} \triangle_i$ satisfies*

$$\psi_{k+1} \geq (1 - 2\omega_{k+1} + Q\omega_{k+1}^2)\psi_k + 14CQ\omega_{k+1}^2 + 4Q\triangle_k \omega_{k+1} \tag{17}$$

Proof. Plug in $\psi_k = \lambda\omega_k + 2Q \sup_{i\geq k_0} \triangle_i$ in equation (17), it's equivalent to

$$(\lambda\omega_{k+1} + 2Q \sup_{i\geq k_0} \triangle_i) \geq (1 - 2\omega_{k+1} + Q\omega_{k+1}^2)(\lambda\omega_k + 2Q \sup_{i\geq k_0} \triangle_i) + 14CQ\omega_{k+1}^2 + 4Q\triangle_k\omega_{k+1}$$

Rearranging terms, we need to show

$$\lambda(\omega_{k+1} - \omega_k + 2\omega_k\omega_{k+1} - Q\omega_k\omega_{k+1}^2) \geq (-2\omega_{k+1} + Q\omega_{k+1}^2)(2Q \sup_{i\geq k_0} \triangle_i) + 14CQ\omega_{k+1}^2 + 4Q\triangle_k\omega_{k+1}$$

Using the fact that $\triangle_k - \sup_{i\geq k_0} \triangle_i < 0$, it is suffices to show that

$$\lambda(C_3 - Q\omega_k)\omega_{k+1}^2 \geq \omega_{k+1}^2(C_4 + 2Q^2 \sup_{i\geq k_0} \triangle_i)$$

where $C_3 = \liminf_{k\to\infty} 2\frac{\omega_k}{\omega_{k+1}} + \frac{\omega_{k+1} - \omega_k}{\omega_{k+1}^2}$, $C_4 = 14CQ^2$. By choosing λ_0 and k_0 such that $\omega_{k_0} \leq \frac{C_3}{2Q}$, and $\lambda_0 = \frac{4Q^2 \sup_{i\geq k_0} \triangle_i + 2C_4}{C_3}$, the desired inequality (17) holds.

□

Theorem 1. *Suppose Assumptions 1-3 hold, the sequence $\{G_k, k = 1, \cdots, \infty\}$ converge to G_*, and there exist a sufficiently large k_0 such that*

$$\mathbb{E}||G_k - G_*||^2 = \mathcal{O}(\lambda\omega_k + \sup_{i\geq k_0} \mathbb{E}||\delta(M, n, \epsilon_i)||)$$

Proof. Denote by $E_k = G_k - G_*$, we have

$$||E_{k+1}||^2 = ||E_k||^2 + \omega_{k+1}^2||\tilde{H}(\boldsymbol{\beta_{k+1}}, G_k)||^2 + 2\omega_{k+1}\mathbb{E}\langle E_k, \tilde{H}(\boldsymbol{\beta_{k+1}}, G_k)\rangle \tag{18}$$

For the third term in (18), we have

$$\begin{aligned}
\langle E_k, \tilde{H}(\boldsymbol{\beta_{k+1}}, G_k)\rangle &\leq \langle E_k, H(\boldsymbol{\beta_{k+1}}, G_k) + \delta(M, n, \epsilon_k)\rangle \\
&\leq \langle E_k, h(G_k) + \mu_{G_k}(\boldsymbol{\beta}_{k+1}) - \Pi_{G_k}\mu_{G_k}(\boldsymbol{\beta}_{k+1}) + \delta(M, n, \epsilon_k)\rangle \\
&\leq -||E_k||^2 + \langle E_k, \mu_{G_k}(\boldsymbol{\beta}_{k+1}) - \Pi_{G_k}\mu_{G_k}(\boldsymbol{\beta}_k)\rangle + \langle E_k, \Pi_{G_k}\mu_{G_k}(\boldsymbol{\beta}_k) - \Pi_{G_{k-1}}\mu_{G_{k-1}}(\boldsymbol{\beta}_k)\rangle \\
&\quad + \langle E_k, \Pi_{G_{k-1}}\mu_{G_{k-1}}(\boldsymbol{\beta}_k) - \Pi_{G_k}\mu_{G_k}(\boldsymbol{\beta}_{k+1})\rangle + ||E_k||||\delta(M, n, \epsilon_k)|| \\
&:= -||E_k||^2 + (\mathrm{I}) + (\mathrm{II}) + (\mathrm{III}) + ||E_k||\triangle_k,
\end{aligned}$$

where we use Lemma 3, Assumption 3, and Cauchy-Schwarz in the second last step, and $||\delta(M, n, \epsilon_k)|| = \triangle_k$.

For (I), we have $\mathbb{E}[\mu_{G_k}(\boldsymbol{\beta}_{k+1}) - \Pi_{G_k}\mu_{G_k}(\boldsymbol{\beta}_k)|\mathcal{F}_k] = 0$, where $\mathcal{F}_k$ is a σ-filter formed by $\{G_0, \boldsymbol{\beta}_1, G_1, \cdots, \boldsymbol{\beta}_k, G_k\}$.

For (II), by Assumption 3

$$\mathbb{E}\langle E_k, \Pi_{G_k}\mu_{G_k}(\boldsymbol{\beta}_k) - \Pi_{G_{k-1}}\mu_{G_{k-1}}(\boldsymbol{\beta}_k)\rangle \leq C||E_k||||G_k - G_{k-1}|| \leq 4CQ^2\omega_k \leq 5CQ^2\omega_{k+1}, \tag{19}$$

where we use the fact that $||G_k - G_{k-1}|| = ||\omega_k\tilde{H}(\boldsymbol{\beta_k}, G_{k-1})|| \leq 2Q\omega_k$, and the last inequality in (19) use the assumption on the step size for a sufficient large number k.

For (III), by Assumption 3

$$\begin{aligned}
&\langle E_k, \Pi_{G_{k-1}}\mu_{G_{k-1}}(\boldsymbol{\beta}_k) - \Pi_{G_k}\mu_{G_k}(\boldsymbol{\beta}_{k+1})\rangle = z_k - z_{k+1} + \langle E_{k+1} - E_k, \Pi_{G_k}\mu_{G_k}(\boldsymbol{\beta}_{k+1})\rangle \\
&\leq z_k - z_{k+1} + C||E_{k+1} - E_k|| = z_k - z_{k+1} + C||G_{k+1} - G_k|| \leq z_k - z_{k+1} + 2CQ\omega_{k+1}
\end{aligned}$$

where $z_k = \langle E_k, \Pi_{G_{k-1}} \mu_{G_{k-1}}(\boldsymbol{\beta}_k)\rangle$, $z_{k+1} = \langle E_{k+1}, \Pi_{G_k} \mu_{G_k}(\boldsymbol{\beta}_{k+1})\rangle$.

Thus,

$$\mathbb{E}||E_{k+1}||^2 \leq (1 - 2\omega_{k+1} + Q\omega_{k+1}^2)\mathbb{E}||E_k||^2 + 14CQ\omega_{k+1}^2 + 4Q\triangle_k\omega_{k+1} + 2\omega_{k+1}\mathbb{E}[z_k - z_{k+1}]\rangle$$

According to Lemma 7, there exists λ_0, k_0 such that

$$\mathbb{E}||E_{k_0}||^2 \leq \psi_{k_0} = \lambda_0\omega_{k_0} + 2Q \sup_{i \geq k_0} \triangle_i$$

Thus,

$$\mathbb{E}||E_k||^2 \leq \psi_k + \mathbb{E}[\sum_{j=k_0+1}^{k} \Lambda_j^k (z_{j+1} - z_j)] \tag{20}$$

From Assumption 3 and Lemma 4, we have

$$\mathbb{E}[|z_k|] = \mathbb{E}\left[\left|\langle E_k, \Pi_{G_{k-1}} \mu_{G_{k-1}}(\boldsymbol{\beta}_k)\rangle\right|\right] \leq \mathbb{E}||E_k||\mathbb{E}\left[\left|\Pi_{G_{k-1}} \mu_{G_{k-1}}(\boldsymbol{\beta}_k)\right|\right] \leq 2QC$$

By Lemma 6,

$$\begin{aligned}\mathbb{E}\left[\left|\sum_{j=k_0+1}^{k} \Lambda_j^k (z_{j+1} - z_j)\right|\right] &= \mathbb{E}\left[\left|\sum_{j=k_0+1}^{k-1} (\Lambda_{j+1}^k - \Lambda_j^k) z_j + \Lambda_{k_0+1}^k z_{k_0} - \Lambda_k^k z_k\right|\right] \\ &\leq (\Lambda_k^k - \Lambda_{k_0+1}^k) 2QC + 8QC\omega_k \leq 12QC\omega_k\end{aligned}$$

Then the inequality (20) can be further bounded as

$$\begin{aligned}\mathbb{E}||E_k||^2 &\leq \lambda_0\omega_k + 2Q \sup_{i \geq k_0} \triangle_i + 12QC\omega_k \\ &= \lambda\omega_k + 2Q \sup_{i \geq k_0} \triangle_i\end{aligned}$$

where $\lambda = \lambda_0 + 12QC$. □

4.2 Weak convergence of model parameters

Given a metric tensor $G(\boldsymbol{\beta}(t))$ on the manifold, the Langevin diffusion is characterized by

$$d\boldsymbol{\beta}(\boldsymbol{t}) = G(\boldsymbol{\beta}(t))\left[\nabla_\beta L(\boldsymbol{\beta}(t)) + \Gamma(\boldsymbol{\beta}(t))\right] + G^{\frac{1}{2}}(\boldsymbol{\beta}(t))d\mathcal{B}_t \tag{21}$$

where $\mathcal{B}_t$ is the standard Brownian motion.

Let $\mathcal{L}$ be the generator for (21), for any function f which is compactly supported and twice differentiable,

$$\mathcal{L}f(\boldsymbol{\beta}(t)) = \left(G(\boldsymbol{\beta}(t))\left[\nabla_\beta L(\boldsymbol{\beta}(t)) + \Gamma(\boldsymbol{\beta}(t))\right] \cdot \nabla_\beta + \frac{1}{2} G^{\frac{1}{2}}(\boldsymbol{\beta}) G^{\frac{1}{2}}(\boldsymbol{\beta})^T : \nabla_\beta \nabla_\beta^T\right) f(\boldsymbol{\beta}(t)) \tag{22}$$

where $\cdot$ denote the vector dot product, and $:$ denote the matrix double dot product, and generator $\mathcal{L}$ is associated with the backward Kolmogorov equation

$$\mathbb{E}[f(\boldsymbol{\beta}(t))] = e^{t\mathcal{L}} f(\boldsymbol{\beta}_0)$$

In our work, we define the true generator using G_* as

$$\mathcal{L}_* = G_* \nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}(t)) \cdot \nabla_{\boldsymbol{\beta}} + \frac{1}{2} G_* : \nabla_{\boldsymbol{\beta}} \nabla_{\boldsymbol{\beta}}^T \tag{23}$$

Given a test function ϕ of interest, let $\bar{\phi}$ be the posterior average of ϕ under the invariant measure of the associate SDE of (23). Let $\boldsymbol{\beta}_k$ be numerical samples, and define $\hat{\phi} = \sum_{k=1}^K \frac{\epsilon_k}{S_K} \phi(\boldsymbol{\beta}_k)$, where $S_K = \sum_{k=1}^K \epsilon_k$. Let ψ be a functional which solves the following Poisson equation

$$\mathcal{L}_* \psi(\boldsymbol{\beta}_k) = \phi(\boldsymbol{\beta}_k) - \bar{\phi}.$$

The solution functional characterize the difference between the posterior average and $\phi(\boldsymbol{\beta}_k)$ for every $\boldsymbol{\beta}_k$. The assumption of ψ is described as follows, which is the same as in [5].

Assumption 4. *The functional ψ, and its derivatives $\mathcal{D}^j \psi$ $(j = 1, 2, 3)$, are bounded by a function $\mathcal{V}$. That is $||\mathcal{D}^j \psi|| \leq C_j \mathcal{V}^{p_j}$ $(j = 0, 1, 2, 3)$, for some positive constants C_j and p_j. Furthermore, $\mathcal{V}$ satisfies $\sup_k \mathbb{E}(\mathcal{V}(\boldsymbol{\beta}_k)) < \infty$, and is smooth such that*

$$\sup_{s \in (0,1)} \mathcal{V}^p \left(s\boldsymbol{\beta} + (1-s)\boldsymbol{\gamma}\right) \leq C \left(\mathcal{V}^p(\boldsymbol{\beta}) + \mathcal{V}^p(\boldsymbol{\gamma})\right)$$

, $\forall \boldsymbol{\beta}, \boldsymbol{\gamma}$, and $p \leq \max\{2p_k\}$, $C > 0$.

Next, we write the local integrator of our proposed method $\tilde{\mathcal{L}}_t$ as

$$\tilde{\mathcal{L}}_k = G(\boldsymbol{\beta}_k) \left(\nabla_{\boldsymbol{\beta}} \tilde{L}(\boldsymbol{\beta}_k)\right) \cdot \nabla_{\boldsymbol{\beta}} + \frac{1}{2} G(\boldsymbol{\beta}_k) : \nabla_{\boldsymbol{\beta}} \nabla_{\boldsymbol{\beta}}^T \tag{24}$$

Then $\tilde{\mathcal{L}}_k = \mathcal{L}_* + \Delta V_k$, with

$$\Delta V_k = \left(G(\boldsymbol{\beta}_k) - G_*\right) \nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}_k) \cdot \nabla_{\boldsymbol{\beta}} + \left(G(\boldsymbol{\beta}_k) - G_*\right) \xi_k \cdot \nabla_{\boldsymbol{\beta}} + \frac{1}{2} \mathrm{tr}\left[\left(G(\boldsymbol{\beta}_k) - G_*\right)^T \nabla_{\boldsymbol{\beta}} \nabla_{\boldsymbol{\beta}}^T\right]$$

where ξ_k is the stochastic noise which comes from $\nabla_{\boldsymbol{\beta}} \tilde{L}(\boldsymbol{\beta}_k) - \nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}_k)$.

We now state the estimates for the bias and MSE.

Theorem 2. *Under Assumptions 4, the bias and MSE of HAMCMC-SA for K steps with decreasing step size ϵ_k is bounded,*

$$\textit{Bias: } |\mathbb{E}\hat{\phi} - \bar{\phi}| = O\left(\frac{1}{S_K} + \sum_{k=1}^K \frac{\lambda \omega_k \epsilon_k}{S_K} + \sum_{k=1}^K \frac{\epsilon_k^2}{S_K} + 2Q \sup_{i \geq k_0} \triangle_i\right)$$

Proof. Following a similar proof as in [5], one can obtain the following:

$$\hat{\phi} - \bar{\phi} = \frac{1}{S_K} \left(\mathbb{E}\psi(\boldsymbol{\beta}_L) - \psi(\boldsymbol{\beta}_0)\right) + \frac{1}{S_K} \sum_{k=1}^{K-1} \left(\mathbb{E}\psi(\boldsymbol{\beta}_k) - \psi(\boldsymbol{\beta}_k)\right) - \sum_{k=1}^K \frac{\epsilon_k}{S_K} \Delta V_k \psi(\boldsymbol{\beta}_{k-1}) + C \frac{\epsilon_k^2}{S_K} \tag{25}$$

Taking expectation on both sides of (25),

$$\left|\mathbb{E}\hat{\phi} - \bar{\phi}\right| \leq \frac{1}{S_K} \mathbb{E}\left|\psi(\boldsymbol{\beta}_K) - \psi(\boldsymbol{\beta}_0)\right| + \sum_{k=1}^K \frac{\epsilon_k}{S_K} \left|\mathbb{E}[\Delta V_k \psi(\boldsymbol{\beta}_{k-1})]\right| + C \sum_{k=1}^K \frac{\epsilon_k^2}{S_K} \tag{26}$$

For the third term in the above equation,

$$|\mathbb{E}\left[\Delta V_k \psi(\boldsymbol{\beta}_{k-1})\right]| \tag{27}$$

$$\leq |\mathbb{E}\langle (G(\boldsymbol{\beta}_k) - G_*)\nabla_\beta K(\boldsymbol{\beta}_k), \nabla_\beta \psi(\boldsymbol{\beta}_{k-1})\rangle| + \frac{1}{2}||G(\boldsymbol{\beta}_k) - G_*||||\mathbb{E}\Delta\psi(\boldsymbol{\beta}_{k-1})|| \tag{28}$$

where we use the fact that $\nabla_\beta \tilde{L}(\boldsymbol{\beta}_k)$ is an unbiased estimator of $\nabla_\beta L(\boldsymbol{\beta}_k)$, and $\mathrm{tr}(AB) = ||AB||_F \leq ||A||_F||A||_F$ where $||\cdot||_F$ is the Frobenius norm and is abbreviate for $||\cdot||$.

According to Assumption 4, we have derivatives $\psi(\boldsymbol{\beta}_{k-1})$ are bounded,

$$\langle (G(\boldsymbol{\beta}_k) - G_*)\nabla_\beta L(\boldsymbol{\beta}_k), \nabla_\beta \psi(\boldsymbol{\beta}_{k-1})\rangle \leq C||G(\boldsymbol{\beta}_k) - G_*||$$

for some positive constant C, since $\nabla_\beta L(\boldsymbol{\beta}_k)$ is also bounded.

By Theorem 1, (27) can be further bounded

$$|\mathbb{E}\left[\Delta V_k \psi(\boldsymbol{\beta}_{k-1})\right]| \leq C\mathbb{E}||G(\boldsymbol{\beta}_k) - G_*|| \leq C(\lambda\omega_k + 2Q\sup_{i\geq k_0}\triangle_i)$$

Thus,

$$\begin{aligned}\left|\mathbb{E}\hat{\phi} - \bar{\phi}\right| &= O\left(\frac{1}{S_K} + \sum_{k=1}^{K}\frac{\epsilon_k}{S_K}(\lambda\omega_k + 2Q\sup_{i\geq k_0}\triangle_i) + \sum_{k=1}^{K}\frac{\epsilon_k^2}{S_K}\right)\\ &= O\left(\frac{1}{S_K} + \sum_{k=1}^{K}\frac{\lambda\omega_k\epsilon_k}{S_K} + \sum_{k=1}^{K}\frac{\epsilon_k^2}{S_K} + 2Q\sup_{i\geq k_0}\triangle_i\right)\end{aligned}$$

where $\omega_k = \mathcal{O}(k^{-\alpha})$. As $K \to \infty$, $\left|\mathbb{E}\hat{\phi} - \bar{\phi}\right| \to 2Q\sup_{i\geq k_0}\triangle_i$, which is a controllable bias.

As for the MSE, we following a similar proof as in [5], as long as $\sup_k \mathbb{E}||\Delta V_k\psi(\boldsymbol{\beta}_{k-1})||^2$ is bounded, which is obvious, we have as $K \to \infty$, $\mathbb{E}\left(\hat{\phi} - \bar{\phi}\right)^2 \to 0$.

□

5 Numerical examples

In the last section, we will perform several numerical tests using the proposed algorithm.

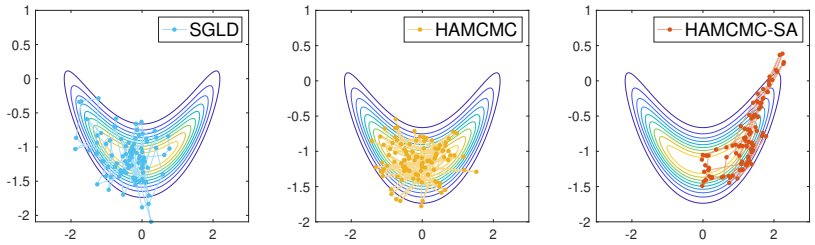
5.1 2D correlated distribution

We first consider a synthetic 2D correlated distribution as shown in [16] for illustration. The density function for the 2D correlated distribution is

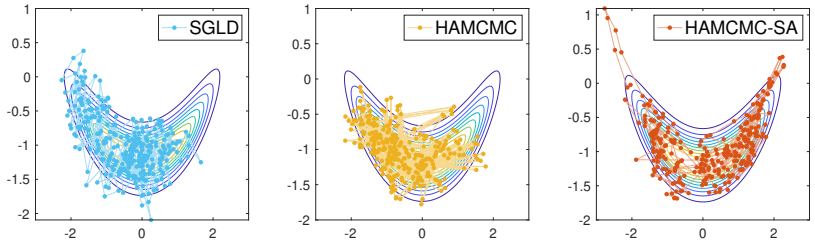
$$p(\beta_1, \beta_2) \propto -\frac{\beta_1^4}{10} - \frac{\left(4(\beta_2 + 1.2) - \beta_1^2\right)^2}{2}.$$

In such a case, the two random variables are highly correlated and the probability distribution of (β_1, β_2) can be easily visualized. We compare the sampling efficiency of the proposed method HAMCMC-SA with vanilla SGLD and HAMCMC. The learning rate are chosen to be set to be $\eta = 0.06, 0.04, 0.06$ for HAMCMC-SA with SGLD and HAMCMC, respectively. The memory size in L-BFGS is chosen to be $M = 3$ for HAMCMC-SA and HAMCMC. The decay rate in the stochastic approximation of Hessian for HAMCMC-SA is $\omega_k = \frac{2}{(10+k)^{0.9}}$. The inverse temperature $\tau = 1$.

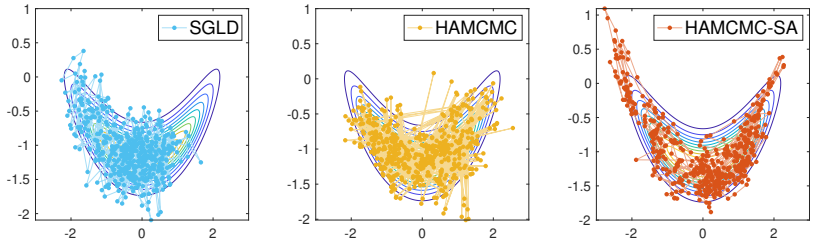
We remark that, in the implementation of HAMCMC, we adopt the idea of approximating inverse Hessian using memory vectors at time steps $\{t-2M+1, \cdots, t-M+1, t-M-1, \cdots, t-1\}$ and updating the current parameter based on the sample at the time step $t-M$, without using the other tricks introduced in [20]. The choices of hyperparameters are consistent in our examples when employing HAMCMC and HAMCMC-SA methods. In Figure 1, we show the sampling trajectories using three methods. The contour of the true posterior is shown in the background. It shows that HAMCMC-SA can explore the posterior better compared to SGLD and HAMCMC.



(a) Number of samples: 100



(b) Number of samples: 300



(c) Number of samples: 500

Figure 1: 2D correlated distribution, sample trajectory.

5.2 Small n large p problem

We then test on a linear regression problem with n observations and p model parameters, where $n << p$. Let the model parameters be $\boldsymbol{\beta} \in \mathbb{R}^p$, $\beta_1 = 3, \beta_2 = 1, \beta_j = 0$, for $j = 3, \cdots, p$. Denote by $X \in \mathbb{R}^{n \times p}$ the predictors, which is generated from $\mathcal{N}_p(0, \Sigma)$ with $\Sigma_{ij} = 0.8^{\frac{1}{4}|i-j|}$. The responses $y = X\boldsymbol{\beta} + \epsilon$, and $\epsilon \sim \mathcal{N}_n(0, 3I_n)$. Due to the error in the observations of y, the originally deterministic parameter $\boldsymbol{\beta}$ results in stochasticity. With our choice of X, the posterior distribution are correlated to demonstrate the performance of the proposed algorithm. In this example, we take $n = 100$ and $p = 200$. We compare the performance of SGLD, HAMCMC and HAMCMC-SA and present them in Figure 2. We remark that, in this example, we assume the model parameter $\boldsymbol{\beta}_j$ follows a

spike and slab Gaussian-Laplace prior in order to perform sparse inference. That is, $\boldsymbol{\beta}_j|\sigma^2, \gamma_j \sim \gamma_j \mathcal{N}(0, \sigma^2 v_1) + (1-\gamma_j)\mathcal{L}(0, \sigma v_0)$, where $\gamma_j = \{0, 1\}$. Similar as in [11], the hyper-parameters priors are $\sigma \sim IG(\nu/2, \nu\lambda/2)$, $\pi(\gamma_j|\delta_j) = \delta_j^{|\gamma_j|}(1-\delta_j)^{p_j - |\gamma_j|}$, and $\pi(\delta_j) = \delta_j^{a-1}(1-\delta_j)^{b-1}$. The priors will be learned through optimization. We choose $\nu = 1, \lambda = 1, v_1 = 100, v_0 = 0.1, \delta = 0.5, a = 1, b = p$, and the step size for updating hyper-parameters in the priors and Hessian approximation is $\omega_k = 2 \times (50 + k)^{-0.75}$. The inverse temperature $\tau = 1$. The learning rates are chosen to be $\eta = 0.1$ for HAMCMC and HAMCMC-SA, and $\eta = 0.001$ for SGLD. The learning rates are different such that the gradients in three method have different magnitude. In our experiments, larger learning rate for SGLD will not lead to convergence. The comparison of posterior mean $\hat{\boldsymbol{\beta}}$ and true $\boldsymbol{\beta}$ is shown in the left subplot of Figure 2. It shows that HAMCMC-SA identifies the model parameters better. Moreover, for testing purposes, we generate 50 new samples, and use the estimated posterior mean in each step to perform a prediction. Then we compute the mean MSE and MAE error of the predicted responses with true responses among these testing samples, and show the results in Figure 2. We observe that HAMCMC-SA has consistently smaller errors during this process.

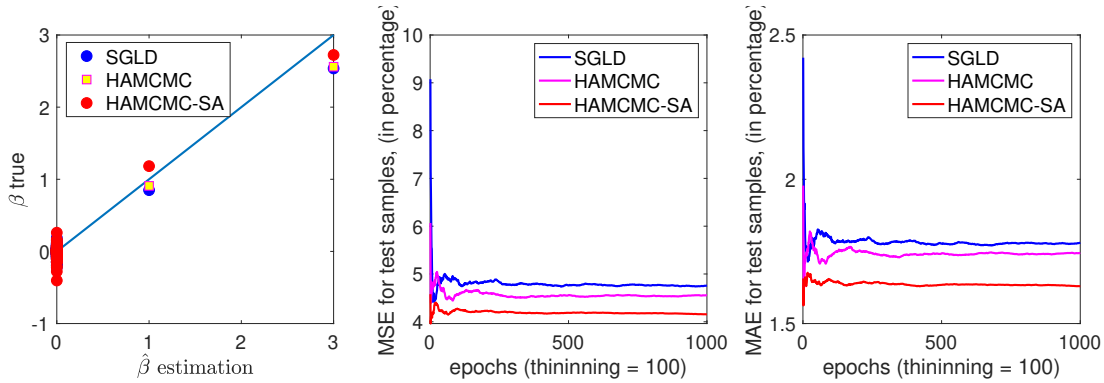


Figure 2: A comparison between three methods for large-p-small-n problem.

5.3 Solutions of Elliptic PDE

Next, we apply the proposed approaches to predict solutions the elliptic problem with heterogeneous permeability fields. The mixed formulation of the elliptic problem reads:

$$\begin{aligned} \kappa^{-1} u + \nabla p &= 0 \quad \text{in} \quad \Omega \\ \text{div}(u) &= f \quad \text{in} \quad \Omega \end{aligned}$$

where κ represents permeability, f is the source. The domain$\Omega = [0, 1] \times [0, 1]$, and the boundary consists of $\partial\Omega = \Gamma_N \cup \Gamma_D$. Raviart-Thomas element RT_0 and piecewise constant element P_0 pairs are chosen to solve the linear system, and the solution vectors will be used as training labels. The mixed finite element system on the fine grid has the matrix form

$$\begin{bmatrix} A_h & B_h^T \\ B_h & 0 \end{bmatrix} \begin{bmatrix} u_h \\ p_h \end{bmatrix} = \begin{bmatrix} D_b \\ -F \end{bmatrix}$$

where where $[A_h]_{ij} = \int_\Omega \kappa^{-1} \psi_i \cdot \psi_j$, and $[B_h]_{ij} = -\int \Omega p_k \, \text{div} \psi_j$, where ψ_j is the velocity basis on the i-th fine scale edge, p_k is the pressure basis on the k-th fine scale block.

It is well known that the multiscale properties of the permeability fields require very fine-scale meshes to recover all scale information. Numerous methods have been proposed to develop reduced-order models to alleviate the computational burden. A popular class of approaches among these includes the mixed multiscale finite element method [7, 8]. The idea is to construct a multiscale velocity basis by solving some local problems on each coarse region and couple them with a mixed formulation. If the underlying permeability has rich information, several multiscale bases are needed to capture these features to provide an accurate approximation. The mixed FEM formulation on the coarse grid level preserves mass conservative property which is essential for flow problems.

To be specific, denote by N_u^H be dimension of the multiscale velocity solution space, and let $R_u \in \mathbb{R}^{N_u^H \times N_u^h}$ be the matrix with these velocity basis in every row, where N_u^h is the dimension of fine scale velocity solution space. Similarly, denote by R_p the matrix containing piecewise constant basis on coarse grid level which maps fine scale pressure vector in $\mathbb{R}^{N_p^h}$ to coarse scale pressure vector in $\mathbb{R}^{N_p^H}$. The mixed formulation on the coarse grid reads

$$\begin{bmatrix} A_H & B_H^T \\ B_H & 0 \end{bmatrix} \begin{bmatrix} u_H \\ p_H \end{bmatrix} = \begin{bmatrix} R_u & 0 \\ 0 & R_p \end{bmatrix} \begin{bmatrix} A_h(\kappa) & B_h^T \\ B_h & 0 \end{bmatrix} \begin{bmatrix} R_u^T & 0 \\ 0 & R_p^T \end{bmatrix} \begin{bmatrix} u_H \\ p_H \end{bmatrix} = \begin{bmatrix} 0 \\ -F_H \end{bmatrix}$$

One can observe that $\begin{bmatrix} R_u & 0 \\ 0 & R_p \end{bmatrix}$ performs an upscaling procedure which is analogy to an encoder, and $\begin{bmatrix} R_u^T & 0 \\ 0 & R_p^T \end{bmatrix}$ acts as downscaling matrix which can be viewed as a decoder.

After one obtains the coarse-scale solution vector u_H from the above system, the multiscale solution u_{ms} can be recovered using $u_{\text{ms}} = \sum_{i=1}^{N_u^H} (u_H)_i \Psi_i$, where $(u_H)_i$ is the i-th component in u_H, and Ψ_i is the i-th column in R_u^T. To obtain an accurate approximation u_{ms} to u_h, it is crucial to design good local problems and basis selecting algorithms that are used for solving multiscale bases. Moreover, many practical applications need to solve the flow problem with (1) varying source terms or boundary conditions, given a fixed permeability field, or (2) different permeability fields. In the second case, the multiscale basis needs to be reconstructed every time providing a new κ. To avoid these technical difficulties, we aim to borrow the upscaling-downscaling idea from coarse grid solvers and construct an encoding-decoding type of neural network [23] as surrogate models (1) between the source term f and fine grid velocity solution u_h, (2) between the permeability fields κ and fine grid velocity solution u_h. We refer to [23] for the details of the network architecture.

5.3.1 Varying source term

we first consider the case when f are different among samples, but the κ is a fixed permeability field from SPE10 model. We use a three-spot source term, where the three blocks with nonzero source lie in the center $\omega_c \in \Omega$, the upper right corner $\omega_{up} \in \Omega$ and lower left corner $\omega_{ll} \in \Omega$ of the computational domain. The values of the source is set to be

$$f(x) = \begin{cases} f_1 \sim \mathcal{N}(10, 5), & \text{if } x \in \omega_{up} \\ f_2 \sim \mathcal{N}(10, 5), & \text{if } x \in \omega_{ll} \\ -(f_1 + f_2), & \text{if } x \in \omega_c \\ 0 & \text{otherwise} \end{cases}$$

An illustration of the permeability field, source term and corresponding velocity solution is shown in Figure 3.

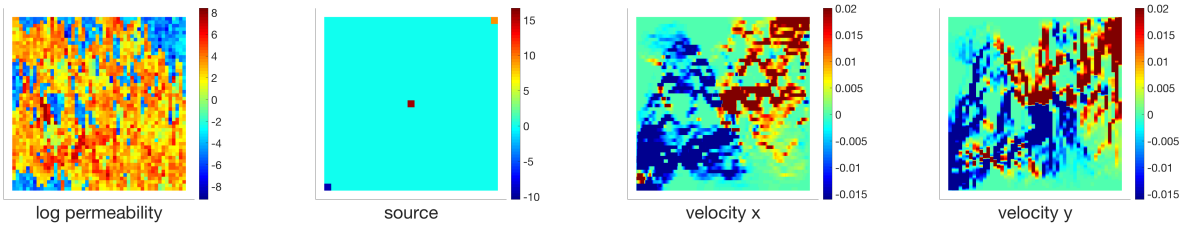


Figure 3: From left to right: The permeability field of SPE10 model (in log scale); A three-spot source; Velocity solution magnitude in x direction; Velocity solution magnitude in y direction.

We simulate 1500 different source terms and use the source-velocity pairs to train the neural network $\mathcal{F}$, where $\mathcal{F}(f) \approx u$. 80% of the samples are randomly selected to train the network and the rest 20% will be used for testing. The learning rate is set to be 0.01. The inverse temperature $\tau = 10000$. The batch size is set to be 100. The decay rate in the stochastic approximation of Hessian for HAMCMC-SA is $\omega_k = \frac{5}{(1000+k)^{0.9}}$. The architecture of the network is as follows. The first layer is an average pooling layer with pool size 2×2, a flatten layer is followed to transform the image into its vector version, then a fully connected layer with 100 neurons is adopted. This part of the network encodes the input and is in analogy to upscaling. Then we reshape this intermediate output to square images, use another two convolution layers, a flatten layer, and a fully connected layer with 200 neurons to extract more hidden features. Finally, a dense layer is used to decode the features. The network has $2,566,828$ weight parameters in total.

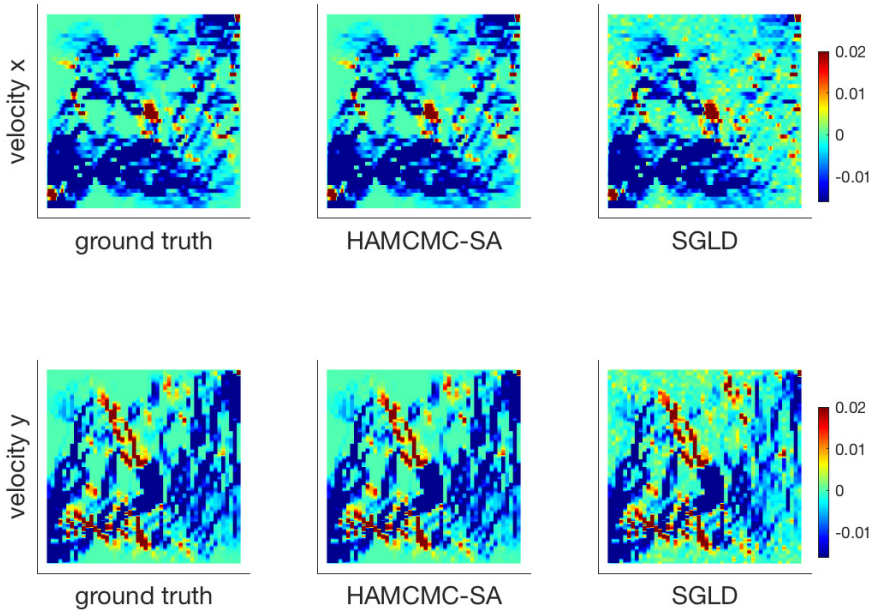
We use the relative l_2 error in the loss function

$$||u_i - \mathcal{F}(f_i)|| = \frac{||u_i - \mathcal{F}(f_i)||_2}{||u_i||_2}$$

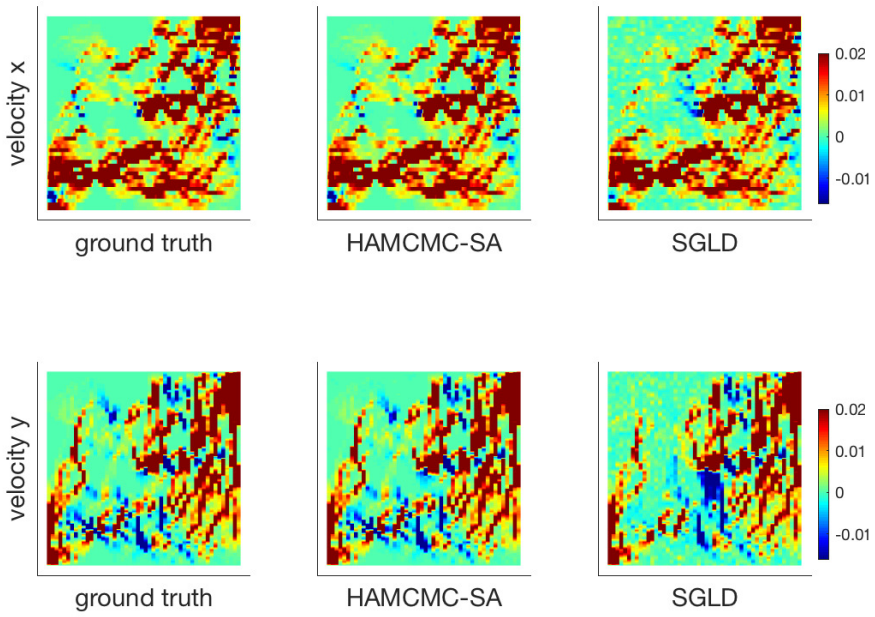
where u_i is the true velocity solution obtained from mixed FEM solver, $\mathcal{F}(f_i)$ is the neural network prediction for the i-th sample. The mean errors for testing are shown in Table 1. We see that with 1 or 2 memory size, HAMCMC-SA gives smaller errors consistently compared with vanilla SGLD. A few sample comparisons are shown in Figure 4. We remark that these are some bad predictions in the testing set, for other sample predictions, the errors are small and the discrepancies cannot be visualized obviously. We observe that, SGLD predictions lose some features compared with true solution, while HAMCMC-SA captures the heterogeneities in the solution well.

	SGLD	HAMCMC-SA (M=1)	HAMCMC-SA (M=2)
No pruning	2.03	0.45	0.42
Pruning Sparse rate 30%	1.38	0.37	0.34
Pruning Sparse rate 50%	1.25	0.29	0.27
Pruning Sparse rate 70%	1.26	0.30	0.27

Table 1: Mean errors (in percentage) for 300 testing samples among the true and predicted solutions using proposed HAMCMC-SA with memory size $M = 1$, $M = 2$, and SGLD.



(a) Test case 1



(b) Test case 2

Figure 4: Varying source term: comparison between true and predicted solution

5.3.2 Varying heterogeneous coefficients

In this section, we consider the case when heterogeneous coefficients vary and let $f = 1$ be a constant source term. The boundary conditions are $u \cdot n = 0$ on the top and bottom sides of the square domain, $p = 1$ on the left boundary, and $p = 0$ on the right boundary.

κ can be obtained using Karhunen-Loeve expansion as follows:

$$\kappa(x;\mu) = \kappa_0 + \sum_{j=1}^{p} \mu_j \sqrt{\xi_j} \Phi_j(x)$$

where κ_0 is a constant which is the mean of the random field. Moreover, random variables μ_j are drawn from i.i.d $N(0,1)$. $(\sqrt{\xi_j}, \Phi_j(x))$ are the eigen-pairs obtained from a Gaussian covariance kernel:

$$\text{Cov}(x_i, y_i; x_j, y_j) = \sigma \exp(-\frac{|x_i - x_j|^2}{l_x^2} - \frac{|y_i - y_j|^2}{l_y^2})$$

where we choose $[l_x, l_y] = [0.2, 0.3]$, $\sigma = 2$ and $p = 64$ in our example.

The training and testing data for deep learning can be generated by solving the equations with MFEM for various permeability fields. An illustrations of the permeability fields for $p = 32, 64, 128$ and corresponding their corresponding solutions are presented in 5. We can see that when p becomes larger, the velocity solutions exhibit many more scale features.

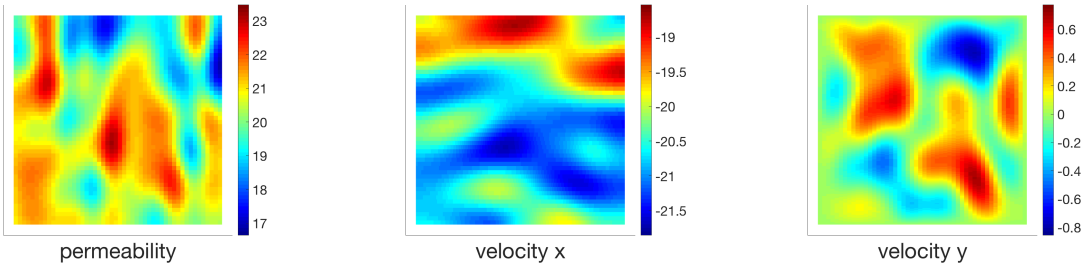


Figure 5: Illustrations of the permeability fields when using 64 terms in KLE expansion and corresponding solutions. From left to right: Permeability, horizontal velocity magnitude, and vertical velocity magnitude.

We generate $1,500$ samples pairs (κ_i, u_h^i), and randomly pick $1,300$ of them for training, and take the rest for testing. In this example, we choose the learning rate to be 0.01. The inverse temperature $\tau = 50000$. The batch size is set to be 100. The decay rate in the stochastic approximation of Hessian for HAMCMC-SA is $\omega_k = \frac{5}{(10+k)^{0.9}}$. The size of an input permeability is 50×50, an output velocity solution vector is $5,100$. The network consists of 2 convolution layers with kernel size 3×3, and 64 and 32 channels, respectively. Then, an average pooling layer with pool size 2×2 is followed by a flatten layer and then a dense layer with 100 neurons. This part of the network can be viewed as an encoder. Then, a reshaping layer, another two convolution layers, a flatten layer, and a fully connected layer with 800 neurons are used to mimic the coarse grid solver. Finally, a fully connected layer is used as a decoder. The total number of parameters is 8,252, and 320.

The numerical results using SGLD and HAMCMC-SA are presented in Table 2. As an illustration, predictions of two samples are presented in Figure 6. The predictions obtained from vanilla SGLD are not reliable, and HAMCMC-SA produces much better results.

	SGLD	HAMCMC-SA (M=1)	HAMCMC-SA (M=2)
No pruning	3.07	2.72	1.68
Pruning Sparse rate 30%	3.04	0.85	0.78
Pruning Sparse rate 50%	3.06	1.21	0.88

Table 2: Mean errors among 300 testing samples between the true and predicted solutions using proposed HAMCMC-SA and SGLD.

6 Conclusion

In this work, we proposed an adaptive Hessian approximated stochastic gradient MCMC method where the parameters are sampled from a posterior lying on a Riemannian manifold. The preconditioning matrix contains the geometric information of the underlying density function and is updated via stochastic approximation in each iteration. It includes an approximation to the inverse Hessian which can be efficiently computed using a limited memory BFGS algorithm. We provide an analysis of the convergence of the proposed method and show that there is a controllable bias introduced by the stochastic approximation. The bias term is generated due to the use of mini-batch when estimating the gradients, and the memory size which is used to approximate the inverse Hessian. It is expected to decrease if the batch size and the memory size are increased and if the step size in stochastic approximation and learning rate is decreased. In practice, our proposed algorithm achieves faster convergence and provides accurate predictions. In the future, we will explore the applications of our proposed method to sparse deep learning.

Acknowledgement

We gratefully acknowledge the support from the National Science Foundation (DMS-1555072, DMS-1736364, CMMI-1634832, and CMMI-1560834), Brookhaven National Laboratory Subcontract 382247, ARO/MURI grant W911NF-15-1-0562 and Department of Energy DE-SC0021142.

References

[1] S. Ahn, A. Korattikara, and M. Welling, *Bayesian posterior sampling via stochastic gradient fisher scoring*, arXiv preprint arXiv:1206.6380, (2012).

[2] A. Benveniste, M. Métivier, and P. Priouret, *Adaptive algorithms and stochastic approximations*, vol. 22, Springer Science & Business Media, 2012.

[3] A. Bordes, L. Bottou, and P. Gallinari, *Sgd-qn: Careful quasi-newton stochastic gradient descent*, Journal of Machine Learning Research, 10 (2009), pp. 1737–1754.

[4] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, *A stochastic quasi-newton method for large-scale optimization*, SIAM Journal on Optimization, 26 (2016), pp. 1008–1031.

[5] C. Chen, N. Ding, and L. Carin, *On the convergence of stochastic gradient mcmc algorithms with high-order integrators.*, In Advances in Neural Information Processing Systems, (2015), pp. 2278–2286.

[6] T. Chen, E. Fox, and C. Guestrin, *Stochastic gradient hamiltonian monte carlo*, in International conference on machine learning, 2014, pp. 1683–1691.

[7] Z. Chen and T. Hou, *A mixed multiscale finite element method for elliptic problems with oscillating coefficients*, Mathematics of Computation, 72 (2002), pp. 541–576.

[8] E. Chung, Y. Efendiev, and C. Lee, *Mixed generalized multiscale finite element methods and applications*, SIAM Multicale Model. Simul., 13 (2014), pp. 338–366.

[9] Y. Dauphin, H. De Vries, and Y. Bengio, *Equilibrated adaptive learning rates for non-convex optimization*, in Advances in neural information processing systems, 2015, pp. 1504–1512.

[10] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, *Identifying and attacking the saddle point problem in high-dimensional non-convex optimization*, in Advances in neural information processing systems, 2014, pp. 2933–2941.

[11] W. Deng, X. Zhang, F. Liang, and G. Lin, *An adaptive empirical bayesian method for sparse deep learning.*, In Advances in Neural Information Processing Systems, (2019), pp. 5564–5574.

[12] N. Ding, Y. Fang, R. Babbush, C. Chen, R. D. Skeel, and H. Neven, *Bayesian sampling using stochastic gradient thermostats*, in Advances in neural information processing systems, 2014, pp. 3203–3211.

[13] 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.

[14] C. Li, C. Chen, D. Carlson, and L. Carin, *Preconditioned stochastic gradient langevin dynamics for deep neural networks.*, In Thirtieth AAAI Conference on Artificial Intelligence, (2016).

[15] D. C. Liu and J. Nocedal, *On the limited memory bfgs method for large scale optimization*, Mathematical programming, 45 (1989), pp. 503–528.

[16] Y.-A. Ma, T. Chen, and E. Fox, *A complete recipe for stochastic gradient mcmc*, in Advances in Neural Information Processing Systems, 2015, pp. 2917–2925.

[17] A. Mokhtari and A. Ribeiro, *Global convergence of online limited memory bfgs*, The Journal of Machine Learning Research, 16 (2015), pp. 3151–3181.

[18] S. Patterson and Y. W. Teh., *Stochastic gradient riemannian langevin dynamics on the probability simplex.*, In Advances in neural information processing systems, (2013), pp. 3102–3110.

[19] H. Robbins and S. Monro, *A stochastic approximation method*, The annals of mathematical statistics, (1951), pp. 400–407.

[20] U. Simsekli, R. Badeau, T. Cemgil, and G. Richard, *Stochastic quasi-newton langevin monte carlo*, 2016.

[21] S. J. Vollmer, K. C. Zygalakis, and Y. W. Teh, *Exploration of the (non-) asymptotic bias and variance of stochastic gradient langevin dynamics*, The Journal of Machine Learning Research, 17 (2016), pp. 5504–5548.

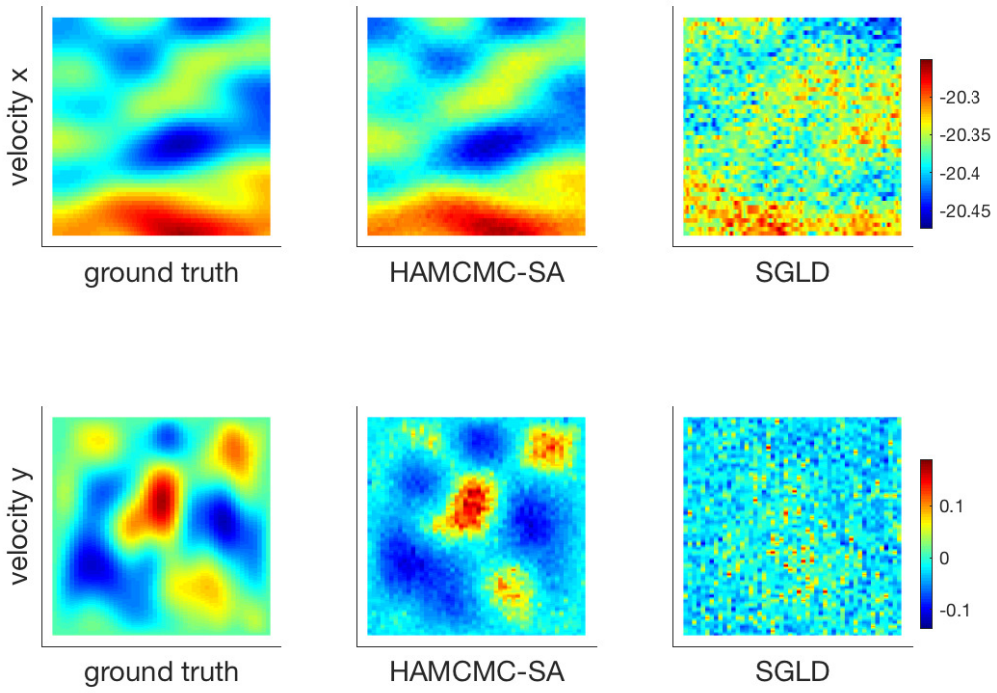
[22] X. Wang, S. Ma, D. Goldfarb, and W. Liu, *Stochastic quasi-newton methods for non-convex stochastic optimization*, SIAM Journal on Optimization, 27 (2017), pp. 927–956.

[23] Y. Wang and G. Lin, *Efficient deep learning techniques for multiphase flow simulation in heterogeneous porousc media.*, Journal of Computational Physics, 401 (2020), p. 108968.

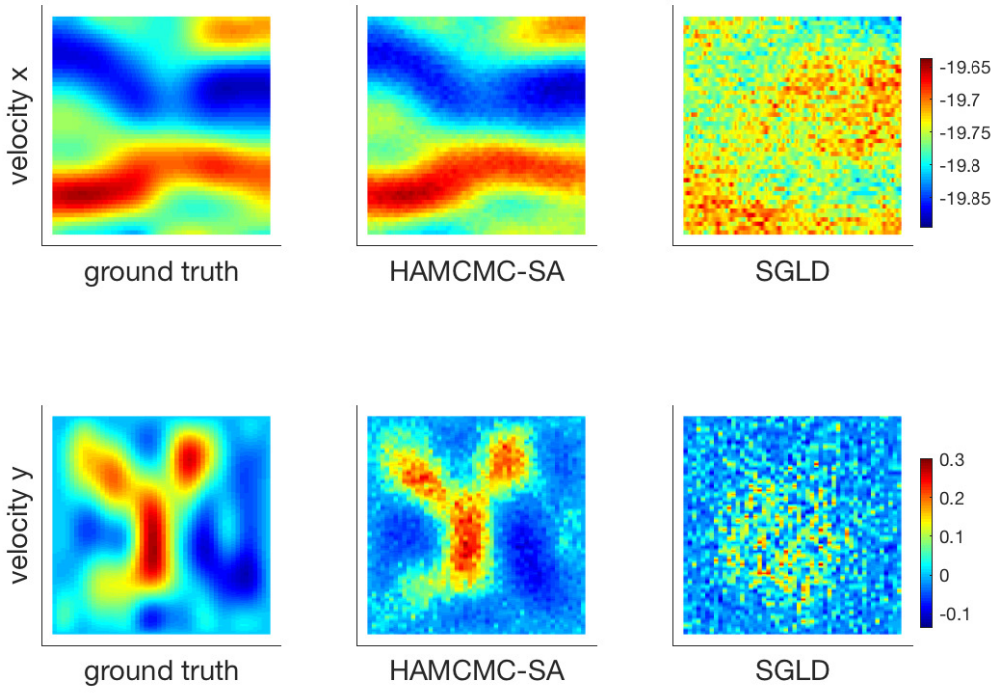
[24] M. Welling and Y. W. Teh, *Bayesian learning via stochastic gradient langevin dynamicsn*, In Proceedings of the 28th international conference on machine learning (ICML-11), (2011), pp. 681–688.

[25] T. Xifara, C. Sherlock, S. Livingstone, S. Byrne, and M. Girolami, *Langevin diffusions and the metropolis-adjusted langevin algorithm*, Statistics & Probability Letters, 91 (2014), pp. 14–19.

[26] 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.



(a) Test case 1



(b) Test case 2

Figure 6: Varying heterogeneous coefficients: comparison between true and predicted solution