# Asynchronous Stochastic Quasi-Newton MCMC for Non-Convex Optimization

**Umut Şimşekli** [1] **Çağatay Yıldız** [2] **Thanh Huy Nguyen** [1] **Gaël Richard** [1] **A. Taylan Cemgil** [3]

## Abstract

Recent studies have illustrated that stochastic gradient Markov Chain Monte Carlo techniques have a strong potential in non-convex optimization, where local and global convergence guarantees can be shown under certain conditions. By building up on this recent theory, in this study, we develop an asynchronous-parallel stochastic L-BFGS algorithm for non-convex optimization. The proposed algorithm is suitable for both distributed and shared-memory settings. We provide formal theoretical analysis and show that the proposed method achieves an ergodic convergence rate of $\mathcal{O}(1/\sqrt{N})$ ($N$ being the total number of iterations) and it can achieve a linear speedup under certain conditions. We perform several experiments on both synthetic and real datasets. The results support our theory and show that the proposed algorithm provides a significant speedup over the recently proposed synchronous distributed L-BFGS algorithm.

## 1. Introduction

Quasi-Newton (QN) methods are powerful optimization techniques that are able to attain fast convergence rates by incorporating local geometric information through an approximation of the inverse of the Hessian matrix. The L-BFGS algorithm (Nocedal & Wright, 2006) is a well-known *limited-memory* QN method that aims at solving the following optimization problem:

$$\theta^\star = \operatorname*{arg\,min}_{\theta \in \mathbb{R}^d} \Big\{ U(\theta) \triangleq \sum_{i=1}^{N_Y} U_i(\theta) \Big\}, \qquad (1)$$

where $U$ is a twice continuously differentiable function that can be convex or non-convex, and is often referred to as the empirical risk. In a typical machine learning context, a dataset $Y$ with $N_Y$ independent and identically distributed (i.i.d.) data points is considered, which renders the function $U$ as a sum of $N_Y$ different functions $\{U_i\}_{i=1}^{N_Y}$.

In large scale applications, the number of data points $N_Y$ often becomes prohibitively large and therefore using a 'batch' L-BFGS algorithm becomes computationally infeasible. As a remedy, *stochastic* L-BGFS methods have been proposed (Byrd et al., 2016; Schraudolph et al., 2007; Moritz et al., 2016; Zhou et al., 2017; Yousefian et al., 2017; Zhao et al., 2017), which aim to reduce the computational requirements of L-BFGS by replacing $\nabla U$ (i.e. the full gradients that are required by L-BFGS) with some *stochastic gradients* that are computed on small subsets of the dataset. However, using stochastic gradients within L-BFGS turns out to be a challenging task since it brings additional technical difficulties, which we will detail in Section 2.

In a very recent study, Berahas et al. (2016) proposed a parallel stochastic L-BFGS algorithm, called multi-batch L-BFGS (mb-L-BFGS), which is suitable for synchronous distributed architectures. This work illustrated that carrying out L-BFGS in a distributed setting introduces further theoretical and practical challenges; however, if these challenges are addressed, stochastic L-BFGS can be powerful in a distributed setting as well, and outperform conventional algorithms such as distributed stochastic gradient descent (SGD), as shown by their experimental results.

Despite the fact that synchronous parallel algorithms have clear advantages over serial optimization algorithms, the computational efficiency of synchronous algorithms is often limited by the overhead induced by the synchronization and coordination among the worker processes. Inspired by asynchronous parallel stochastic optimization techniques (Agarwal & Duchi, 2011; Lian et al., 2015; Zhang et al., 2015; Zhao & Li, 2015; Zheng et al., 2017), in this study, we propose an asynchronous parallel stochastic L-BFGS algorithm for large-scale non-convex optimization problems. The proposed approach aims at speeding up the synchronous algorithm presented in (Berahas et al., 2016) by allowing all the workers work independently from each

[1]LTCI, Télécom ParisTech, Université Paris-Saclay, 75013, Paris, France [2]Department of Computer Science, Aalto University, Espoo, 02150, Finland [3]Department of Computer Engineering, Boğaziçi University, 34342, Bebek, Istanbul, Turkey. Correspondence to: Umut Şimşekli <umut.simsekli@telecom-paristech.fr>.

*Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

other and circumvent the inefficiencies caused by synchronization and coordination.

Extending stochastic L-BFGS to asynchronous settings is a highly non-trivial task and brings several challenges. In our strategy, we first reformulate the optimization problem (1) as a sampling problem where the goal becomes drawing random samples from a distribution whose density is concentrated around $\theta^\star$. We then build our algorithm upon the recent stochastic gradient Markov Chain Monte Carlo (SG-MCMC) techniques (Chen et al., 2015; 2016b) that have close connections with stochastic optimization techniques (Dalalyan, 2017; Raginsky et al., 2017; Zhang et al., 2017), and have proven successful in large-scale Bayesian machine learning. We provide formal theoretical analysis and prove non-asymptotic guarantees for the proposed algorithm. Our theoretical results show that the proposed algorithm achieves an ergodic global convergence with rate $\mathcal{O}(1/\sqrt{N})$, where $N$ denotes the total number of iterations. Our results further imply that the algorithm can achieve a linear speedup under ideal conditions.

For evaluating the proposed method, we conduct several experiments on synthetic and real datasets. The experimental results support our theory: our experiments on a large-scale matrix factorization problem show that the proposed algorithm provides a significant speedup over the synchronous parallel L-BFGS algorithm.

## 2. Technical Background

**Preliminaries:** As opposed to the classical optimization perspective, we look at the optimization problem (1) from a *maximum a-posteriori* (MAP) estimation point of view, where we consider $\theta$ as a random variable in $\mathbb{R}^d$ and $\theta^\star$ as the optimum of a Bayesian posterior whose density is given as $p(\theta|Y) \propto \exp(-U(\theta))$, where $Y \equiv \{Y_1, \dots, Y_{N_Y}\}$ is a set of i.i.d. observed data points. Within this context, $U(\theta)$ is often called the *potential energy* and defined as $U(\theta) = -[\log p(\theta) + \sum_{i=1}^{N_Y} \log p(Y_i|\theta)]$, where $p(Y_i|\theta)$ is the likelihood function and $p(\theta)$ is the prior density. In a classical optimization context, $-\log p(Y_i|\theta)$ would correspond to the data-loss and $-\log p(\theta)$ would correspond to a regularization term. Throughout this study, we will assume that the problem (1) has a unique solution in $\mathbb{R}^d$.

We define a stochastic gradient $\nabla\tilde{U}(\theta)$, that is an unbiased estimator of $\nabla U$, as follows: $\nabla\tilde{U}(\theta) = -[\nabla \log p(\theta) + \frac{N_Y}{N_\Omega}\sum_{i\in\Omega} \nabla \log p(Y_i|\theta)]$, where $\Omega \subset \{1, \dots, N_Y\}$ denotes a random data subsample that is drawn with replacement, $N_\Omega = |\Omega|$ is the cardinality of $\Omega$. In the sequel, we will occasionally use the notation $\nabla\tilde{U}_n$ and $\nabla\tilde{U}_\Omega$ to denote the stochastic gradient computed at iteration $n$ of a given algorithm, or on a specific data subsample $\Omega$, respectively.

**The L-BFGS algorithm:** The L-BFGS algorithm iteratively applies the following equation in order to find the MAP estimate given in (1):

$$\theta_n = \theta_{n-1} - hH_n\nabla U(\theta_{n-1}) \tag{2}$$

where $n$ denotes the iterations. Here, $H_n$ is an approximation to the inverse Hessian at $\theta_{n-1}$ and is computed by using the $M$ past values of the 'iterate differences' $s_n \triangleq \theta_n - \theta_{n-1}$, and 'gradient differences' $y_n \triangleq \nabla U(\theta_n) - \nabla U(\theta_{n-1})$. The collection of the iterate and gradient differences is called the *L-BFGS memory*. The matrix-vector product $H_n\nabla U(\theta_{n-1})$ is often implemented by using the *two-loop recursion* (Nocedal & Wright, 2006), which has linear time and space complexities $\mathcal{O}(Md)$.

In order to achieve computational scalability, stochastic L-BFGS algorithms replace $\nabla U$ with $\nabla\tilde{U}$. This turns out to be problematic, since the gradient differences $y_n$ would be *inconsistent*, meaning that the stochastic gradients in different iterations will be computed on different data subsamples, i.e. $\Omega_{n-1}$ and $\Omega_n$. On the other hand, in the presence of the stochastic gradients, L-BFGS is no longer guaranteed to produce positive definite approximations even in convex problems, therefore more considerations should be taken in order to make sure that $H_n$ is positive definite.

**Stochastic Gradient Markov Chain Monte Carlo:** Along with the recent advances in MCMC techniques, diffusion-based algorithms have become increasingly popular due to their applicability in large-scale machine learning applications. These techniques, so called the Stochastic Gradient MCMC (SG-MCMC) algorithms, aim at generating samples from the posterior distribution $p(\theta|Y)$ as opposed to finding the MAP estimate, and have strong connections with stochastic optimization techniques (Dalalyan, 2017). In this line of work, Stochastic Gradient Langevin Dynamics (SGLD) (Welling & Teh, 2011) is one of the pioneering algorithms and generates an approximate sample $\theta_n$ from $p(\theta|Y)$ by iteratively applying the following update equation:

$$\theta_n = \theta_{n-1} - h\nabla\tilde{U}_n(\theta_{n-1}) + \sqrt{2h/\beta}Z_n \tag{3}$$

where $h$ is the step-size and $\{Z_n\}_{n=1}^N$ is a collection of standard Gaussian random variables in $\mathbb{R}^d$. Here, $\beta$ is called the *inverse temperature*: it is fixed to $\beta = 1$ in vanilla SGLD and when $\beta \neq 1$ the algorithm is called 'tempered'. In an algorithmic sense, SGLD is identical to SGD, except that it injects a Gaussian noise at each iteration and it coincides with SGD when $\beta$ goes to infinity.

SGLD has been extended in several directions (Ma et al., 2015; Chen et al., 2015; Şimşekli et al., 2016b; Şimşekli, 2017). In (Şimşekli et al., 2016a), we proposed an L-BFGS-based SGLD algorithm with $\mathcal{O}(M^2d)$ computational complexity, which aimed to improve the convergence speed of the vanilla SGLD. We showed that a straightforward way of combining L-BFGS in SGLD would incur an undesired bias; however, the remedy to prevent this bias resulted in numerical instability, which would limit the applicability of the algorithm. In other recent studies, SGLD has also been extended to synchronous (Ahn et al., 2014) and asynchronous (Chen et al., 2016b; Springenberg et al., 2016) distributed MCMC settings.

SGLD can be seen as a discrete-time simulation of a continuous-time Markov process that is the solution of the following stochastic differential equation (SDE):

$$d\theta_t = -\nabla U(\theta_t)dt + \sqrt{2/\beta}dW_t, \tag{4}$$

where $W_t$ denotes the standard Brownian motion in $\mathbb{R}^d$. Under mild regularity conditions on $U$, the solution process $(\theta_t)_{t\geq 0}$ attains a unique stationary distribution with a density that is proportional to $\exp(-\beta U(\theta))$ (Roberts & Stramer, 2002). An important property of this distribution is that, as $\beta$ goes to infinity, this density concentrates around the global minimum of $U(\theta)$ (Hwang, 1980; Gelfand & Mitter, 1991). Therefore, for large enough $\beta$, a random sample that is drawn for the stationary distribution of $(\theta_t)_{t\geq 0}$ would be close to $\theta^\star$. Due to this property, SG-MCMC methods have recently started drawing attention from the non-convex optimization community. Chen et al. (2016a) developed an annealed SG-MCMC algorithm for non-convex optimization and it was recently extended by Ye et al. (2017). Raginsky et al. (2017) and Xu et al. (2017) provided finite-time guarantees for SGLD to find an 'approximate' global minimizer that is close to $\theta^\star$, which imply that the additive Gaussian noise in SGLD can help the algorithm escape from poor local minima. In a complementary study, Zhang et al. (2017) showed that SGLD enters a neighborhood of a local minimum of $U(\theta)$ in polynomial time, which shows that even if SGLD fails to find the global optimum, it will still find a point that is close to one of the local optima. Even though these results showed that SG-MCMC is promising for optimization, it is still not clear how an asynchronous stochastic L-BFGS method could be developed within an SG-MCMC framework.

## 3. Asynchronous Stochastic L-BFGS

In this section, we propose a novel asynchronous L-BFGS-based (tempered) SG-MCMC algorithm that aims to provide an approximate optimum that is close to $\theta^\star$ by generating samples from a distribution that has a density that is proportional to $\exp(-\beta U(\theta))$. We call the proposed algorithm asynchronous parallel stochastic L-BFGS (as-L-BFGS). Our method is suitable for both distributed and shared-memory settings. We will describe the algorithm only for the distributed setting; the shared-memory version is almost identical to the distributed version as long as the updates are ensured to be *atomic*.

We consider a classical asynchronous optimization architecture, which is composed of a *master node*, several *worker nodes*, and a *data server*. The main task of the master node is to maintain the newest iterate of the algorithm. At each iteration, the master node receives an *additive update vector* from a worker node, it adds this vector to the current iterate in order to obtain the next iterate, and then it sends the new iterate to the worker node which has sent the update vector. On the other hand, the worker nodes work in a completely asynchronous manner. A worker node receives the iterate from the master node, computes an update vector, and sends the update vector to the master node. However, since the iterate would be possibly modified by another worker node which runs asynchronously in the mean time, the update vector that is sent to the server will thus be computed on an *old* iterate, which causes both practical and theoretical challenges. Such updates are aptly called 'delayed' or 'stale'. The full data is kept in the data server and we assume that all the workers have access to the data server.

The proposed algorithm iteratively applies the following update equations in the master node:

$$u_{n+1} = u_n + \Delta u_{n+1}, \qquad \theta_{n+1} = \theta_n + \Delta\theta_{n+1}, \tag{5}$$

where $n$ is the iteration index, $u_n$ is called the *momentum* variable, and $\Delta u_{n+1}$ and $\Delta\theta_{n+1}$ are the update vectors that are computed by the worker nodes. A worker node runs the following equations in order to compute the update vectors:

$$\begin{aligned}\Delta u_{n+1} \triangleq & -h' H_{n+1}(\theta_{n-l_n})\nabla\tilde{U}_{n+1}(\theta_{n-l_n}) - \gamma' u_{n-l_n} \\ & + \sqrt{2h'\gamma'/\beta}Z_{n+1}, \end{aligned} \tag{6}$$

$$\Delta\theta_{n+1} \triangleq H_{n+1}(\theta_{n-l_n})u_{n-l_n}, \tag{7}$$

where $h'$ is the step-size, $\gamma' > 0$ is the *friction* parameter that determines the weight of the momentum, $\beta$ is the inverse temperature, $\{Z_n\}_n$ denotes standard Gaussian random variables, and $H_n$ denotes the L-BFGS matrix at iteration $n$. Here, $l_n \geq 0$ denotes the 'staleness' of a particular update and measures the delay between the current update and the up-to-date iterate that is stored in the master node. We assume that the delays are bounded, i.e. $\max_n l_n \leq l_{\max} < \infty$. Note that the matrix-vector products have $\mathcal{O}(Md)$ time-space complexity.

Due to the asynchrony, the stochastic gradients and the L-BFGS matrices will be computed on the delayed variables $\theta_{n-l_n}$ and $u_{n-l_n}$. As opposed to the asynchronous stochastic gradient algorithms, where the main difficulty stems from the delayed gradients, our algorithm faces further challenges since it is not straightforward to obtain the gradient and iterate differences that are required for the L-BFGS computations in an asynchronously parallel setting.

**Algorithm 1:** as-L-BFGS: Master node

```
1  input: $\theta_0$, $u_0$
   // Global iteration index
2  $n \leftarrow 0$
3  Send $(\theta_0, u_0)$ to all the workers $w = 1, \dots, W$
4  while $n < N$ do
5      Receive $(\Delta\theta_{n+1}, \Delta u_{n+1})$ from worker $w$
       // Generate the new iterates
6      $u_{n+1} = u_n + \Delta u_{n+1}, \quad \theta_{n+1} = \theta_n + \Delta\theta_{n+1}$
7      Send the iterates $(\theta_{n+1}, u_{n+1})$ to worker $w$
8      Set $n \leftarrow n + 1$
```

We propose the following approach for the computation of the L-BFGS matrices. As opposed to the mb-L-BFGS algorithm, which uses a central L-BFGS memory (i.e. the collection of the gradient and iterate differences) that is stored in the master node, we let each worker have their own local L-BFGS memories since the master node would not be able to keep track of the gradient and iterate differences, which are received in an asynchronous manner. In our strategy, each worker updates its own L-BFGS memory right after sending the update vector to the master node. The overall algorithm is illustrated in Algorithms 1 and 2 ($W$ denotes the number of workers).

In order to be able to have consistent gradient differences, each worker applies a multi-batch subsampling strategy that is similar to mb-L-BFGS. We divide the data subsample into two subsets, i.e. $\Omega_n = \{S_n, O_n\}$ with $N_S \triangleq |S_n|$, $N_O \triangleq |O_n|$, and $N_\Omega = N_S + N_O$. Here the main idea is to choose $N_S \gg N_O$ and use $O_n$ as an overlapping subset for the gradient differences. In this manner, in addition to the gradients that are computed on $S_n$ and $O_n$, we also perform an extra gradient computation on the previous overlapping subset, at the end of each iteration. As $N_O$ will be small, this extra cost will not be significant. Finally, in order to ensure the L-BFGS matrices are positive definite, we use a 'cautious' update mechanism that is useful for non-convex settings (Li & Fukushima, 2001; Zhang & Sutton, 2011; Berahas et al., 2016) as shown in Algorithm 2.

Note that, in addition to asynchrony, the proposed algorithm also extends the current stochastic L-BFGS methods by introducing momentum. This brings two critical practical features: (i) without the existence of the momentum variables, the injected Gaussian noise must depend on the L-BFGS matrices, as shown in (Şimşekli et al., 2016a), which results in an algorithm with $\mathcal{O}(M^2 d)$ time complexity whereas our algorithm has $\mathcal{O}(Md)$ time complexity, (ii) the use of the momentum significantly repairs the numerical instabilities caused by the asynchronous updates, since $u_n$ inherently encapsulates a direction for $\theta_n$, which provides additional information to the algorithm besides the gradients and L-BFGS computations. Furthermore, in a

**Algorithm 2:** as-L-BFGS: Worker node $(w)$

```
1  input: $M, \gamma, N_S, N_O$ $(N_\Omega = N_S + N_O)$
   // Local iteration index
2  $i \leftarrow 0$
3  while the master node is running do
4      Receive $(\theta_{n-l_n}, u_{n-l_n})$ from the master
5      Draw a subsample $\Omega_{n+1} = \{S_{n+1}, O_{n+1}\}$
       // Gradient computation
6      $\nabla\tilde{U}_{n+1}(\theta_{n-l_n}) = \frac{N_O}{N_\Omega}\nabla\tilde{U}_{O_{n+1}}(\theta_{n-l_n}) + \frac{N_S}{N_\Omega}\nabla\tilde{U}_{S_{n+1}}(\theta_{n-l_n})$
7      Compute $(\Delta\theta_{n+1}, \Delta u_{n+1})$ by (6) and (7)
8      Send $(\Delta\theta_{n+1}, \Delta u_{n+1})$ to the master
       // Local variables for L-BFGS
9      $\tilde{\theta}_i = \theta_{n-l_n}, \quad \tilde{O}_i = O_{n+1}, \quad \tilde{g}_i = \nabla\tilde{U}_{O_{n+1}}(\theta_{n-l_n})$
10     if $i \geq 1$ then
           // Compute the overlapping gradient
11         $g' = \nabla\tilde{U}_{\tilde{O}_{i-1}}(\tilde{\theta}_i)$
           // Compute the L-BFGS variables
12         $s_i = \tilde{\theta}_i - \tilde{\theta}_{i-1}, \quad y_i = g' - \tilde{g}_{i-1}$
           // Cautious memory update
13         Add $(s_i, y_i)$ to the L-BFGS memory only if
               $y_i^\top s_i \geq \epsilon\|s_i\|^2$ for some $\epsilon > 0$
14     Set $i \leftarrow i + 1$
```

very recent study (Loizou & Richtárik, 2017) the use of momentum variables has been shown to be useful in other second-order optimization methods. On the other hand, despite their advantages, the momentum variable also drifts apart the proposed algorithm from the original L-BFGS formulation. However, even such approximate approaches have proven useful in various scenarios (Zhang & Sutton, 2011; Fu et al., 2016). Also note that, when $\beta \to \infty$, $l_{\max} = 0$, and $H_n(\theta) = I$ for all $n$, the algorithm coincides with SGD with momentum. A more detailed illustration is given in the supplementary document.

## 4. Theoretical Analysis

In this section, we will provide non-asymptotic guarantees for the proposed algorithm. Our analysis strategy is different from the conventional analysis approaches for stochastic optimization and makes use of tools from analysis of SDEs. In particular, we will first develop a continuous-time Markov process whose marginal stationary measure admits a density that is proportional to $\exp(-\beta U(\theta))$. Then we will show that (5)-(7) form an approximate Euler-Maruyama integrator that approximately simulates this continuous process in discrete-time. Finally, we will analyze this approximate numerical scheme and provide a non-asymptotic error bound. All the proofs are given in the supplementary document.

We start by considering the following stochastic dynamical system:

$$dp_t = \Big[\frac{1}{\beta}\Gamma_t(\theta_t) - H_t(\theta_t)\nabla_\theta U(\theta_t) - \gamma p_t\Big]dt + \sqrt{\frac{2\gamma}{\beta}}dW_t$$
$$d\theta_t = H_t(\theta_t)p_t dt \tag{8}$$

where $p_t \in \mathbb{R}^d$ is also called the *momentum* variable, $H_t(\cdot)$ denotes the L-BFGS matrix at time $t$ and $\Gamma_t(\cdot)$ is a vector that is defined as follows:

$$\Big[\Gamma_t(\theta)\Big]_i \triangleq \sum_{j=1}^{d} \frac{\partial [H_t(\theta)]_{ij}}{\partial [\theta]_j}, \tag{9}$$

where $[v]_i$ denotes the $i^{\text{th}}$ component of a vector $v$ and similarly $[M]_{ij}$ denotes a single element of a matrix $M$.

In order to analyze the invariant measure of the SDE defined in (8), we need certain conditions to hold. First, we have two regularity assumptions on $U$ and $H_t$:

**H1.** *The gradient of the potential is Lipschitz continuous, i.e.* $\|\nabla_\theta U(\theta) - \nabla_\theta U(\theta')\| \leq L\|\theta - \theta'\|$, $\forall \theta, \theta' \in \mathbb{R}^d$.

**H2.** *The L-BFGS matrices have bounded second-order derivatives and they are Lipschitz continuous, i.e.* $\|H_t(\theta) - H_t(\theta')\| \leq L_H\|\theta - \theta'\|$, $\forall \theta, \theta' \in \mathbb{R}^d, t \geq 0$.

The assumptions **H**1 and **H**2 are standard conditions in analysis of SDEs (Duan, 2015) and similar assumptions have also been considered in stochastic gradient (Moulines & Bach, 2011) and stochastic L-BFGS algorithms (Zhou et al., 2017). Besides, **H**2 provides a direct control on the partial derivatives of $H_t$, which will be useful for analyzing the overall numerical scheme. We now present our first result that establishes the invariant measure of the SDE (8).

**Proposition 1.** *Assume that the conditions **H**1 and 2 hold. Let $X_t = [\theta_t^\top, p_t^\top]^\top \in \mathbb{R}^{2d}$ and $(X_t)_{t\geq 0}$ be a Markov process that is a solution of the SDE given in (8). Then $(X_t)_{t\geq 0}$ has a unique invariant measure $\pi$ that admits a density $\rho(X) \propto \exp(-\mathcal{E}(X))$ with respect to the Lebesgue measure, where $\mathcal{E}$ is an energy function on the extended state space and is defined as: $\mathcal{E}(X) \triangleq \beta U(\theta) + \frac{\beta}{2}p^\top p$.*

This result shows that, if the SDE (8) could be exactly simulated, the *marginal* distribution of the samples $\theta_t$ would converge to a measure $\pi_\theta$ which has a density that is proportional to $\exp(-\beta U(\theta))$. Therefore, for large enough $\beta$ and $t$, $\theta_t$ would be close to the global optimum $\theta^\star$.

We note that when $\beta = 1$, the SDE (8) shares similarities with the SDEs presented in (Fu et al., 2016; Ma et al., 2015). While the main difference being the usage of the tempering scheme, (Fu et al., 2016) further differs from our approach as it directly discard the term $\Gamma_t$ since is in a Metropolis-Hastings framework, which is not adequate for large-scale applications. On the other hand, the stochastic gradient Riemannian Hamiltonian Monte Carlo algorithm given in (Ma et al., 2015), chooses $H_t$ as the Fisher information matrix; a quantity that requires $\mathcal{O}(d^2)$ space-time complexity and is not analytically available in general.

We will now show that the proposed algorithm (5)-(7) form an approximate method for simulating (8) in discrete-time. For illustration, we first consider the Euler-Maruyama integrator for (8), given as follows:

$$p_{n+1} = p_n - hH_n(\theta_n)\nabla_\theta U(\theta_n) - h\gamma p_n + \frac{h}{\beta}\Gamma_n(\theta_n)$$
$$+ \sqrt{2h\gamma/\beta}Z_{n+1}, \tag{10}$$
$$\theta_{n+1} = \theta_n + hH_n(\theta_n)p_n. \tag{11}$$

Here, the term $(1/\beta)\Gamma_n$ introduces an additional computational burden and its importance is very insignificant (i.e. its magnitude is of order $\mathcal{O}(1/\beta)$ due to **H**2). Therefore, we discard $\Gamma_n$, define $u_n \triangleq hp_n$, $\gamma' \triangleq h\gamma$, $h' \triangleq h^2$, and use these quantities in (10) and (11). We then obtain the following re-parametrized Euler integrator:

$$u_{n+1} = u_n - h'H_n(\theta_n)\nabla_\theta U(\theta_n) - \gamma' u_n + \sqrt{2h'\gamma'/\beta}Z_{n+1}$$
$$\theta_{n+1} = \theta_n + H_n(\theta_n)u_n$$

The detailed derivation is given in the supplementary document. Finally, we replace $\nabla U$ with the stochastic gradients, replace the variables $\theta_n$ and $u_n$ with stale variables $\theta_{n-l_n}$ and $p_{n-l_n}$ in the update vectors, and obtain the ultimate update equations, given in (5). Note that, due to the negligence of $\Gamma_n$, the proposed approach would require a large $\beta$ and would not be suitable for classical posterior sampling settings, where $\beta = 1$.

In this section, we will analyze the *ergodic* error $\mathbb{E}[\hat{U}_N - U^\star]$, where we define $\hat{U}_N \triangleq (1/N)\sum_{n=1}^{N} U(\theta_n)$ and $U^\star \triangleq U(\theta^\star)$. This error resembles the bias of a statistical estimator; however, as opposed to the bias, it directly measures the expected discrepancy to the global optimum. Similar ergodic error notions have been considered in the analysis of non-convex optimization methods (Lian et al., 2015; Chen et al., 2016a; Berahas et al., 2016).

In our proof strategy, we decompose the error into two terms: $\mathbb{E}[\hat{U}_N - U^\star] = \mathcal{A}_1 + \mathcal{A}_2$, where $\mathcal{A}_1 \triangleq \mathbb{E}[\hat{U}_N - \bar{U}_\beta]$ $\mathcal{A}_2 \triangleq [\bar{U}_\beta - U^\star] \geq 0$, and $\bar{U}_\beta \triangleq \int_{\mathbb{R}^d} U(\theta)\pi_\theta(d\theta)$. We then upper-bound these terms separately.

The term $\mathcal{A}_1$ turns out to be the bias of a statistical estimator, which we can analyze by using ideas from recent SG-MCMC studies. However, existing tools cannot be directly used because of the additional difficulties introduced by the L-BFGS matrices. In order to bound $\mathcal{A}_1$, we first require the following smoothness and boundedness condition.

**H3.** *Let $\psi$ be a functional that is the unique solution of a*

*Poisson equation that is defined as follows:*

$$\mathcal{L}_n \psi(X_n) = U(\theta_n) - \bar{U}_\beta, \quad (12)$$

*where $X_n = [\theta_n^\top, p_n^\top]^\top$, $\mathcal{L}_n$ is the generator of (8) at $t = nh$ and is formally defined in the supplementary document. The functional $\psi$ and its up to third-order derivatives $\mathcal{D}^k\psi$ are bounded by a function $V(X)$, such that $\|\mathcal{D}^k\psi\| \leq C_k V^{r_k}$ for $k = 0, 1, 2, 3$ and $C_k, r_k > 0$. Furthermore, $\sup_n \mathbb{E}V^r(X_n) < \infty$ and $V$ is smooth such that $\sup_{s\in(0,1)} V^r(sX + (1-s)X') \leq C(V^r(X) + V^r(X'))$ for all $X, X' \in \mathbb{R}^{2d}$, $r \leq \max 2r_k$, and $C > 0$.*

Assumption **H**3 is also standard in SDE analysis and SG-MCMC (Mattingly et al., 2010; Teh et al., 2016; Chen et al., 2015; Durmus et al., 2016) and gives us control over the weak error of the numerical integrator. We further require the following regularity conditions in order to have control over the error induced by the delayed stochastic gradients.

**H4.** *The variance of the stochastic gradients is bounded, i.e. $\mathbb{E}\|\nabla_\theta U(\theta) - \nabla_\theta \tilde{U}(\theta)\|^2 \leq \sigma$ for some $0 < \sigma < \infty$.*

**H5.** *For a smooth and bounded function $f$, the remainder $r_{\mathcal{L}_n,f}(\cdot)$ in the following Taylor expansion is bounded:*

$$e^{h\mathcal{L}_n} f(X) = f(X) + h\mathcal{L}_n f(X) + h^2 r_{\mathcal{L}_n,f}(X). \quad (13)$$

The following lemma presents an upper-bound for $\mathcal{A}_1$.

**Lemma 1.** *Assume the conditions **H**1-5 hold. We have the following bound for the bias:*

$$\left|\mathbb{E}[\hat{U}_N - \bar{U}_\beta]\right| = \mathcal{O}\Big(\frac{1}{Nh} + \max(l_{max}, 1)h + \frac{1}{\beta}\Big). \quad (14)$$

Here, the term $1/\beta$ in (14) appears due to the negligence of $\Gamma_n$. In order to bound the second term $\mathcal{A}_2$, we follow a similar strategy to (Raginsky et al., 2017), where we use **H** 1 and the following moment condition on $\pi_\theta$.

**H6.** *The second-order moments of $\pi_\theta$ are bounded and satisfies the following inequality: $\int_{\mathbb{R}^d} \|\theta\|^2 \pi_\theta(d\theta) \leq \frac{C_\beta}{\beta}$, for some $C_\beta > \max(\beta d/(2\pi e), de/L)$.*

This assumption is mild since $\pi_\theta$ concentrates around $\theta^\star$ as $\beta$ tends to infinity. The order $1/\beta$ is arbitrary, hence the assumption can be further relaxed. The following lemma establishes an upper-bound for $\mathcal{A}_2$.

**Lemma 2.** *Under assumptions **H**1 and 6, the following bound holds: $\bar{U}_\beta - U^\star = \mathcal{O}(1/\beta)$.*

We now present our main result, which can be easily proven by combining Lemmas 1 and 2.

**Theorem 1.** *Assume that the conditions **H**1-6 hold. Then the ergodic error of the proposed algorithm is bounded as follows:*

$$\left|\mathbb{E}\hat{U}_N - U^\star\right| = \mathcal{O}\Big(\frac{1}{Nh} + \max(1, l_{max})h + \frac{1}{\beta}\Big). \quad (15)$$

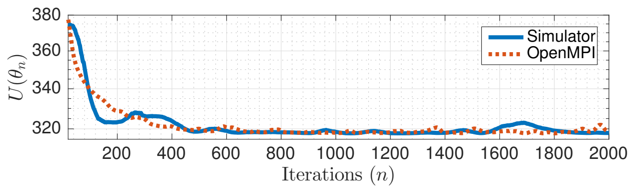

*Figure 1.* The comparison of the simulated and the real implementation of as-L-BFGS with $W = 40$ workers.

More explicit constants and a discussion on the relation of the theorem to other recent theoretical results are provided in the supplementary document.

Theorem 1 provides a non-asymptotic guarantee for convergence to a point that is close to the global optimizer $\theta^\star$ even when $U$ is non-convex, thanks to the additive Gaussian noise. The bound suggests an optimal rate of convergence of $\mathcal{O}(1/\sqrt{N})$, which is in line with the current rates of the non-convex asynchronous algorithms (Lian et al., 2015). Furthermore, if we assume that the total number of iterations $N$ is a linear function of the number of workers, e.g. $N = N_W W$, where $N_W$ is the number of iterations executed by a single worker, Theorem 1 implies that, in the ideal case, the proposed algorithm can achieve a linear speedup with increasing $W$, provided that $l_{\max} = \mathcal{O}(1/(Nh^2))$.

Despite their nice theoretical properties, it is well-known that tempered sampling approaches also often get stuck near a local minimum. In our case, this behavior would be mainly due to the hidden constant in (14), which can be exponential in dimension $d$, as illustrated in (Raginsky et al., 2017) for SGLD. On the other hand, Theorem 1 does not guarantee that the proposed algorithm will converge to a neighborhood of a local minimum; however, we believe that we can also prove local convergence guarantees by using the techniques provided in (Zhang et al., 2017; Tzen et al., 2018), which we leave as a future work.

## 5. Experiments

The performance of asynchronous stochastic gradient methods has been evaluated in several studies, where the advantages and limitations have been illustrated in various scenarios, to name a few (Dean et al., 2012; Zhang et al., 2015; Zheng et al., 2017). In this study, we will explore the advantages of using L-BFGS in an asynchronous environment. In order to illustrate the advantages of asynchrony, we will compare as-L-BFGS with mb-L-BFGS (Berahas et al., 2016); and in order to illustrate the advantages that are brought by using higher-order geometric information, we will compare as-L-BFGS to asynchronous SGD (a-SGD) (Lian et al., 2015). We will also explore the speedup behavior of as-L-BFGS for increasing $W$.

We conduct experiments on both synthetic and real

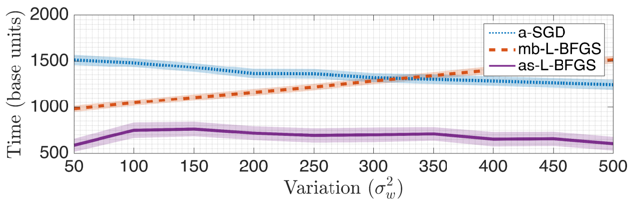

*Figure 2.* The required time to achieve $\varepsilon$-accuracy in the synthetic setting. Solid lines represent the average results and the shades represent three standard deviations.

datasets. For real data experiments, we have implemented all the three algorithms in C++ by using a low-level message passing protocol for parallelism, namely the OpenMPI library. This code can be used both in a distributed environment or a single computer with multiprocessors. For the experiments on synthetic data, we have implemented the algorithms in MATLAB, by developing a realistic *discrete-event simulator*. This simulated environment is particularly useful for understanding the behaviors of the algorithms in detail since we can explicitly control the computation time that is spent at the master or worker nodes, and the communication time between the nodes. This simulation strategy also enables us to explicitly control the variation among the computational powers of the worker nodes; a feature that is much harder to control in real distributed environments.

**Linear Gaussian model:** We conduct our first set of experiments on synthetic data where we consider a rather simple convex quadratic problem whose optimum is analytically available. The problem is formulated as finding the MAP estimate of the following linear Gaussian probabilistic model:

$$\theta \sim \mathcal{N}(0, I), \quad Y_i|\theta \sim \mathcal{N}(a_i^\top \theta, \sigma_x^2), \quad \forall i = 1, \dots, N_Y.$$

We assume that $\{a_n\}_{n=1}^N$ and $\sigma_x^2$ are known and we aim at computing $\theta^\star$. For these experiments, we develop a parametric discrete event simulator that aims to simulate the algorithms in a controllable yet realistic way. The simulator simulates a distributed optimization algorithm once it is provided four parameters: (i) $\mu_m$: the average computational time spent by the master node at each iteration, (ii) $\mu_w$: the average computational time spent by a single worker at each iteration, (iii) $\sigma_w$: the standard deviation of the computational time spent by a single worker per iteration, and (iv) $\tau$: the time spent for communications per iteration. All these parameters are in a generic *base time unit*. Once these parameters are provided for one of the three algorithms, the simulator simulates the (a)synchronous distributed algorithm by drawing random computation times from a log-normal distribution whose mean and variance is specified by $\mu_w$ and $\sigma_w^2$. Figure 1 illustrates a typical outcome of the real and the simulated implementations of as-L-BFGS, where we observe that the simulator is able to provide realistic simulations that can even very well reflect the fluctuations of the algorithm.

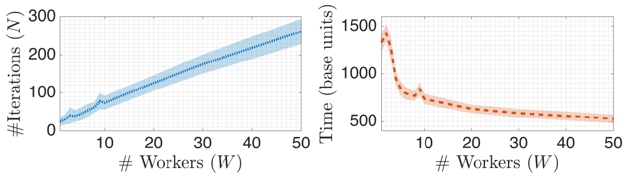

*Figure 3.* The required time to achieve $\varepsilon$-accuracy by as-L-BFGS for increasing number of workers. Solid lines represent the average results and the shades represent three standard deviations.

In our first experiment, we set $d = 100$, $\sigma_x^2 = 10$, $N_Y = 600$, we randomly generate and fix the vectors $\{a_n\}_n$ in such a way that there will be a strong correlation in the posterior distribution, and we finally generate a true $\theta$ and the observations $Y$ by using the generative model.

For each algorithm, we fix $\mu_m$, $\mu_s$, and $\tau_s$ to realistic values and investigate the effect of the variation among the workers by comparing the running time of the algorithms for achieving $\varepsilon$-accuracy (i.e., $(U(\theta_n) - U^\star)/U^\star \leq \varepsilon$) for different values of $\sigma_w^2$ when $W = 40$. We repeat each experiment 100 times. In all our experiments, we have tried several values for the hyper-parameters of each algorithm and we report the best results. All the hyper-parameters are provided in the supplementary document.

Figure 2 visualizes the results for the first experiment. We can observe that, for smaller values $\sigma_w^2$ as-L-BFGS and mb-L-BFGS perform similarly, where a-SGD requires more computational time to achieve $\varepsilon$-accuracy. However, as we increase the value of $\sigma_w^2$, mb-L-BFGS requires more computational time in order to be able to collect sufficient amount of stochastic gradients. The results show that both asynchronous algorithms turn out to be more robust to the variability of the computational power of the workers, where as-L-BFGS shows a better performance.

In our second experiment, we investigate the speedup behavior of as-L-BFGS within the simulated setting. In this setting, we consider a highly varying set of workers and set $\sigma_w^2 = 200$ and vary the number of workers $W$. As illustrated in Figure 3, as $W$ increases, $l_{\max}$ increases as well and the algorithm hence requires more iterations in order to achieve $\varepsilon$-accuracy, since a smaller step-size needs to be used. However, this increment in the number of iterations is compensated by the increased number of workers, as we observe that the required computational time gracefully decreases with increasing $W$. We observe a similar behavior for different values of $\sigma_w^2$, where the speedup is more prominent for smaller $\sigma_w^2$.

**Large-scale matrix factorization:** In our next set of experiments, we consider a large-scale matrix factorization problem (Gemulla et al., 2011; Şimşekli et al., 2015; Şimşekli et al., 2017), where the goal is to obtain the

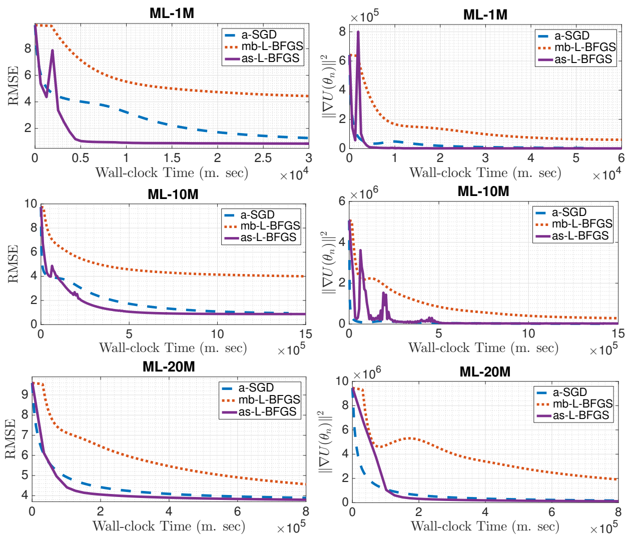

*Figure 4.* The convergence behavior of the algorithms on the MovieLens datasets for $W = 10$.

MAP solution of the following probabilistic model: $F_{rk} \sim \mathcal{N}(0,1)$, $G_{ks} \sim \mathcal{N}(0,1)$, $Y_{rs}|F,G \sim \mathcal{N}\big(\sum_k F_{rk}G_{ks}, 1\big)$. Here, $Y \in \mathbb{R}^{R\times S}$ is the data matrix, and $F \in \mathbb{R}^{R\times K}$ and $G \in \mathbb{R}^{K\times S}$ are the factor matrices to be estimated.

In this context, we evaluate the algorithms on three large-scale movie ratings datasets, namely MovieLens 1Million (ML-1M), 10Million (ML-10M), and 20Million (ML-20M) (`grouplens.org`). The ML-1M dataset contains 1 million non-zero entries, where $R = 3883$ (movies) and $S = 6040$ (users). The ML-10M dataset contains 10 million non-zero entries, resulting in a $10681 \times 71567$ data matrix. Finally, the ML-20M dataset contains 20 million ratings, resulting in a $27278 \times 138493$ data matrix. We have conducted these experiments on a cluster of more than 500 interconnected computers, each of which is equipped with variable quality CPUs and memories. In these experiments, we have found that the numerical stability is improved when $H_n$ is replaced with $(H_n + \rho I)$ for small $\rho > 0$. This small modification does not violate our theoretical results. The hyper-parameters are provided in the supplementary document.

Figure 4 shows the performance of the three algorithms on the MovieLens datasets in terms of the root-mean-squared-error (RMSE), which is a standard metric for recommendation systems, and the norm of the gradients through iterations. In these experiments, we set $K = 5$ for all the three datasets and we set the number of workers to $W = 10$. The results show that, in all datasets, as-L-BFGS provides a significant speedup over mb-L-BFGS thanks to asynchrony. We can observe that even when the speed of convergence of mb-L-BFGS is comparable to a-SGD and as-L-BFGS (cf. the plots showing the norm of the gradients), the final RMSE yielded by mb-L-BFGS is poorer than the two other methods, which is an indicator that the asynchronous algorithms are able to find a better local minimum. On the other hand, the asynchrony causes more fluctuations in as-L-BFGS when compared to a-SGD.

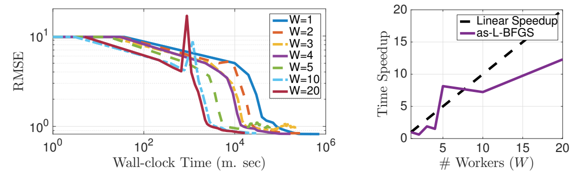

*Figure 5.* The convergence behavior of as-L-BFGS on the ML-1M dataset for increasing number of workers. The 'time speedup' is computed as the ratio of the running time with 1 worker to the running time of $W$ workers.

As opposed to the synthetic data experiments, in all the three MovieLens datasets, we observe that as-L-BFGS provides a slight improvement in the convergence speed when compared to a-SGD. This indicates that a-SGD is able to achieve a comparable convergence speed by taking more steps while as-L-BFGS is computing the matrix-vector products. However, this gap can be made larger by considering a more efficient, yet more sophisticated implementation for L-BFGS computations (Chen et al., 2014).

In our last experiment, we investigate the speedup properties of as-L-BFGS in the real distributed setting. In this experiment, we only consider the ML-1M dataset and run the as-L-BFGS algorithm for different number of workers. Figure 5 illustrates the results of this experiment. As we increase $W$ from 1 to 10, we obtain a decent speedup that is close to a linear speedup. However, when we set $W = 20$ the algorithm becomes unstable, since the term $l_{\max}h$ in (15) dominates. Therefore, for $W = 20$ we need to decrease the step-size $h$, which requires the algorithm to be run for a longer amount of time in order to achieve the same error as we achieved when $W$ was smaller. On the other hand, the algorithm achieves a linear speedup in terms of iterations; however, the corresponding result is provided in the supplementary document due to the space constraints.

## 6. Conclusion

In this study, we proposed an asynchronous parallel L-BFGS algorithm for non-convex optimization. We developed the algorithm within the SG-MCMC framework, where we reformulated the problem as sampling from a concentrated probability distribution. We proved non-asymptotic guarantees and showed that as-L-BFGS achieves an ergodic global convergence with rate $\mathcal{O}(1/\sqrt{N})$ and it can achieve a linear speedup. Our experiments supported our theory and showed that the proposed algorithm provides a significant speedup over the synchronous parallel L-BFGS algorithm.

## Acknowledgments

The authors would like to thank to Murat A. Erdoğdu for fruitful discussions. This work is partly supported by the French National Research Agency (ANR) as a part of the FBIMATRIX project (ANR-16-CE23-0014), by the Scientific and Technological Research Council of Turkey (TÜBİTAK) grant number 116E580, and by the industrial chair Machine Learning for Big Data from Télécom ParisTech.

## References

Agarwal, A. and Duchi, J. C. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems*, pp. 873–881, 2011.

Ahn, S., Shahbaba, B., and Welling, M. Distributed stochastic gradient MCMC. In *International conference on machine learning*, pp. 1044–1052, 2014.

Berahas, A. S., Nocedal, J., and Takác, M. A multi-batch L-BFGS method for machine learning. In *Advances in Neural Information Processing Systems*, pp. 1055–1063, 2016.

Byrd, R. H., Hansen, S. L., Nocedal, J., and Singer, Y. A stochastic quasi-Newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

Chen, C., Ding, N., and Carin, L. On the convergence of stochastic gradient MCMC algorithms with high-order integrators. In *Advances in Neural Information Processing Systems*, pp. 2269–2277, 2015.

Chen, C., Carlson, D., Gan, Z., Li, C., and Carin, L. Bridging the gap between stochastic gradient MCMC and stochastic optimization. In *AISTATS*, 2016a.

Chen, C., Ding, N., Li, C., Zhang, Y., and Carin, L. Stochastic gradient MCMC with stale gradients. In *Advances In Neural Information Processing Systems*, pp. 2937–2945, 2016b.

Chen, W., Wang, Z., and Zhou, J. Large-scale L-BFGS using MapReduce. In *Advances in Neural Information Processing Systems*, pp. 1332–1340, 2014.

Şimşekli, U., Badeau, R., Cemgil, A. T., and Richard, G. Stochastic quasi-Newton Langevin Monte Carlo. In *ICML*, 2016a.

Şimşekli, U., Badeau, R., Richard, G., and Cemgil, A. T. Stochastic thermodynamic integration: efficient Bayesian model selection via stochastic gradient MCMC. In *ICASSP*, 2016b.

Şimşekli, U., Durmus, A., Badeau, R., Richard, G., Moulines, E., and Cemgil, A. T. Parallelized stochastic gradient Markov Chain Monte Carlo algorithms for non-negative matrix factorization. In *ICASSP*, 2017.

Dalalyan, A. S. Further and stronger analogy between sampling and optimization: Langevin Monte Carlo and gradient descent. *Proceedings of the 2017 Conference on Learning Theory*, 2017.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., and Ng, A. Y. Large scale distributed deep networks. In *Advances in neural information processing systems*, pp. 1223–1231, 2012.

Duan, J. *An Introduction to Stochastic Dynamics*. Cambridge University Press, New York, 2015.

Durmus, A., Şimşekli, U., Moulines, E., Badeau, R., and Richard, G. Stochastic gradient Richardson-Romberg Markov Chain Monte Carlo. In *NIPS*, 2016.

Fu, T., Luo, L., and Zhang, Z. Quasi-Newton Hamiltonian Monte Carlo. In *UAI*, 2016.

Gelfand, S. B. and Mitter, S. K. Recursive stochastic algorithms for global optimization in Rˆd. *SIAM Journal on Control and Optimization*, 29(5):999–1018, 1991.

Gemulla, R., Nijkamp, E., J., H. P., and Sismanis, Y. Large-scale matrix factorization with distributed stochastic gradient descent. In *ACM SIGKDD*, 2011.

Hwang, C. Laplace's method revisited: weak convergence of probability measures. *The Annals of Probability*, pp. 1177–1182, 1980.

Li, D.-H. and Fukushima, M. On the global convergence of the BFGS method for nonconvex unconstrained optimization problems. *SIAM Journal on Optimization*, 11 (4):1054–1064, 2001.

Lian, X., Huang, Y., Li, Y., and Liu, J. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pp. 2737–2745, 2015.

Loizou, N. and Richtárik, P. Momentum and stochastic momentum for stochastic gradient, Newton, proximal point and subspace descent methods. *arXiv preprint arXiv:1712.09677*, 2017.

Ma, Y. A., Chen, T., and Fox, E. A complete recipe for stochastic gradient MCMC. In *Advances in Neural Information Processing Systems*, pp. 2899–2907, 2015.

Mattingly, J. C., Stuart, A. M., and Tretyakov, M. V. Convergence of numerical time-averaging and stationary measures via Poisson equations. *SIAM Journal on Numerical Analysis*, 48(2):552–577, 2010.

Moritz, P., Nishihara, R., and Jordan, M. A linearly-convergent stochastic L-BFGS algorithm. In *Artificial Intelligence and Statistics*, pp. 249–258, 2016.

Moulines, E. and Bach, F. R. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pp. 451–459, 2011.

Nocedal, J. and Wright, S. J. *Numerical optimization*. Springer, Berlin, 2006.

Raginsky, M., Rakhlin, A., and Telgarsky, M. Non-convex learning via stochastic gradient Langevin dynamics: a nonasymptotic analysis. In *Proceedings of the 2017 Conference on Learning Theory*, volume 65, pp. 1674–1703, 2017.

Roberts, G. O. and Stramer, O. Langevin Diffusions and Metropolis-Hastings Algorithms. *Methodology and Computing in Applied Probability*, 4(4):337–357, December 2002. ISSN 13875841.

Schraudolph, N. N., Yu, J., and Günter, S. A stochastic quasi-Newton method for online convex optimization. In *Artificial Intelligence and Statistics*, pp. 436–443, 2007.

Şimşekli, U. Fractional Langevin Monte carlo: Exploring Levy driven stochastic differential equations for Markov chain Monte Carlo. In *ICML*, pp. 3200–3209, 2017.

Şimşekli, U., Koptagel, H., Güldaş, H., Cemgil, A. T., Öztoprak, F., and Birbil, Ş. İ. Parallel stochastic gradient Markov Chain Monte Carlo for matrix factorisation models. *arXiv preprint arXiv:1506.01418*, 2015.

Springenberg, J. T., Klein, A., Falkner, S., and Hutter, F. Asynchronous stochastic gradient MCMC with elastic coupling. *arXiv preprint arXiv:1612.00767*, 2016.

Teh, Y. W., Thiery, A. H., and Vollmer, S. J. Consistency and fluctuations for stochastic gradient Langevin dynamics. *Journal of Machine Learning Research*, 17:1–33, 2016.

Tzen, B., Liang, T., and Raginsky, M. Local optimality and generalization guarantees for the langevin algorithm via empirical metastability. In *Proceedings of the 2018 Conference on Learning Theory*, 2018.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient Langevin dynamics. In *International Conference on Machine Learning*, pp. 681–688, 2011.

Xu, P., Chen, J., and Gu, Q. Global convergence of Langevin dynamics based algorithms for nonconvex optimization. *arXiv preprint arXiv:1707.06618*, 2017.

Ye, N., Zhu, Z., and Mantiuk, R. Langevin dynamics with continuous tempering for training deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 618–626. 2017.

Yousefian, F., Nedić, A., and Shanbhag, U. On stochastic and deterministic quasi-Newton methods for non-strongly convex optimization: convergence and rate analysis. *arXiv preprint arXiv:1710.05509*, 2017.

Zhang, S., Choromanska, A. E., and LeCun, Y. Deep learning with elastic averaging sgd. In *Advances in Neural Information Processing Systems*, pp. 685–693, 2015.

Zhang, Y. and Sutton, C. A. Quasi-Newton methods for Markov Chain Monte Carlo. In *Advances in Neural Information Processing Systems*, pp. 2393–2401, 2011.

Zhang, Y., Liang, P., and Charikar, M. A hitting time analysis of stochastic gradient langevin dynamics. In *Proceedings of the 2017 Conference on Learning Theory*, volume 65, pp. 1980–2022, 2017.

Zhao, R., Haskell, W. B., and Tan, V. Y. F. Stochastic L-BFGS: Improved convergence rates and practical acceleration strategies. *IEEE Transactions on Signal Processing*, 2017.

Zhao, S.-Y. and Li, W.-J. Fast asynchronous parallel stochastic gradient decent. *arXiv preprint arXiv:1508.05711*, 2015.

Zheng, S., Meng, Q., Wang, T., Chen, W., Yu, N., Ma, Z., and Liu, T. Asynchronous stochastic gradient descent with delay compensation. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 4120–4129, 2017.

Zhou, C., Gao, W., and Goldfarb, D. Stochastic adaptive quasi-Newton methods for minimizing expected values. In *International Conference on Machine Learning*, pp. 4150–4159, 2017.