J. Appl. Numer. Optim. 8 (2026), No. 2, pp. 157-179
Available online at http://jano.biemdas.com
https://doi.org/10.23952/jano.8.2026.2.01

A NEW MINI-BATCH STOCHASTIC QUASI-NEWTON METHOD FOR DEEP LEARNING

JOHANNES JAHN

Department Mathematik, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 91058 Erlangen, Germany

Abstract. The numerical method presented in this paper combines a mini-batch stochastic gradient method with a simple quasi-Newton method, and it can be used for solving large-scale finite-sum minimization problems. As a consequence of the fact that this particular quasi-Newton method is designed to operate with a simplified matrix class, it is well-suited to implementation on GPUs, thereby enabling its use in the context of deep learning. The number of iterations performed per mini-batch is not merely one, but rather a relatively small number. This quasi-Newton method is examined in detail with regard to the question of superlinear convergence. Moreover, convergence results are provided for the expectation of the distance between the iteration points and the solution of the optimization problem throughout the entire procedure. Even without convexity assumptions, we can show a result regarding the decrease of the objective function values. Finally, the results of numerical tests with a very large number of variables demonstrate the performance of the new algorithm. This method has great potential for applications in deep learning.

Keywords. Deep learning; Mini-batch stochastic gradient methods; Quasi-Newton methods.
2020 Mathematics Subject Classification. 90C06, 90C53, 68T07.

1. INTRODUCTION

Quasi-Newton methods represent a powerful tool for the solution of nonlinear unconstrained optimization problems. In general, these methods have a superlinear rate of convergence under appropriate assumptions. However, for finite-sum minimization problems in deep learning with a very large number of variables and a very high number of data instances in the complete dataset, quasi-Newton methods are not the first choice because matrix-vector multiplications cannot be handled effectively on GPUs, which are the workhorses in AI computing. On the other hand, the class of stochastic gradient methods is the preferred choice in deep learning due to its effective use of GPUs, despite the high number of iterations and relatively slow convergence rate.

The aim of this paper is to combine the advantages of both approaches in a single methodology. The primary methodology employed is a mini-batch stochastic approach. The use of a mini-batch stochastic method allows for the reduction of gradient noise by selecting an appropriate mini-batch size that is not too small. Given the increased mathematical complexity of the objective function of the resulting subproblem, a finite number of iteration steps of a

E-mail address: johannes.jahn@fau.de
Received 24 July 2025; Accepted 26 September 2025; Published online 1 May 2026.

quasi-Newton method are employed to approximate a solution of this subproblem. However, a standard quasi-Newton method is not employed; instead, a special simple variant is utilized for which computations on GPUs make sense. This special quasi-Newton method only works with a simple approximation of the Hessian of the objective function at the solution point, whereby the entries of the iterated matrices are assumed to be equal outside of the diagonal. The structure of the second partial derivatives of the objective function is of paramount importance for the computation of a new iteration point through the use of straightforward formulas.

Although quasi-Newton methods do not necessitate the utilization of a step size, the methodology presented herein permits the incorporation of step sizes, thereby facilitating a more adaptable approach to the influence of gradient noise. In particular, gradient noise represents a significant contributing factor to the poor convergence properties observed in conventional stochastic gradient methods.

In the research literature, there are papers on various quasi-Newton approaches around the proposed method. Zhao, Lai, and Lim [1] proposed a stochastic Steffensen method that combines a mini-batch method with a Steffensen method [2] (compare also [3]) that achieves a high convergence rate only with first order information, and a Barzilai-Borwein procedure [4] whose step size is based on a certain quasi-Newton equation. And Becker, Fadili, and Ochs [5, Alg. 7] also gave a quasi-Newton approach with a Barzilai-Borwein [4] step size rule. However, our paper does not use a Barzilai-Borwein step size rule but rather employs their approach with respect to the quasi-Newton equation.

Another algorithm combining a quasi-Newton method with a stochastic mini-batch procedure was published by Griffin, Jahani, Takáč, Yektamaram, and Zhou [6]. They use a trust region framework together with a quasi-Newton method embedded in a mini-batch scheme. In contrast to our approach, this paper works with a special quasi-Newton update formula, which requires some computational effort.

An interesting recent overview of stochastic quasi-Newton methods was given by Guo, Liu, and Han [7]. The focus is on BFGS-based methods in a stochastic setting. But our way tries to get away from time-consuming matrix updates. The authors point out [7, p. 254] that the convergence rate of quasi-Newton methods in a stochastic setting is generally sublinear. This is due to noisy gradient estimations. Therefore, the reduction of gradient noise is an important task.

An overview of noise reduction methods was given by Bottou, Curtis, and Nocedal [8, Section 5], as part of a very comprehensive overview of methods for large-scale machine learning. They discuss three classes of noise reduction methods, namely dynamic sampling methods, gradient aggregation methods, and iterate averaging methods. All three classes have their own merits. Dynamic sampling methods reduce gradient noise by increasing the size of the mini-batch. Gradient aggregation methods store gradient estimates, work with an update rule, and use a weighted average of these estimates as an iteration direction. Iterate averaging methods use an average of iteration points calculated during the optimization process.

Increasing the size of the mini-batches during the iteration process automatically reduces the stochasticity of the method. Since a quasi-Newton method generally has better convergence properties than a gradient method, an increase in the mini-batch size certainly makes sense, even if we have to expect a higher computational effort. In our paper, we work with a fixed but

not too small mini-batch size, which can be chosen in a way that the numerical effort is not too high.

Further investigations on a combination of quasi-Newton methods and stochastic gradient methods can be found in the papers [9–14]. For specific applications, see also [15–17], and for further research in border areas we refer to [18–22].

This paper is organized as follows: Preliminaries are summarized in Section 2. The algorithm and the derivation of the formulas used are presented in Section 3. Convergence results are the topic of Section 4. The superlinear convergence of the quasi-Newton method used for the subproblems is characterized, and convergence results are given for the complete algorithmic approach. The last section is devoted to numerical results using two illustrative examples.

2. Preliminaries

Throughout this paper we work in $\mathbb{R}^n$ space with normally high dimension $n \in \mathbb{N}$. Our standard assumption is as follows.

Assumption 2.1. For some $p \in \mathbb{N}$, let the functions $f_1, \ldots, f_p : \mathbb{R}^n \to \mathbb{R}$ be given, and let $f : \mathbb{R}^n \to \mathbb{R}$ with

$$f(x) := \frac{1}{p} \sum_{k=1}^{p} f_k(x) \text{ for all } x \in \mathbb{R}^n$$

denote the objective function of the following optimization problem.

Here p plays the role of the dataset size, i.e. p is the number of all data instances within the entire dataset. In general, p is expected to be very large. Under Assumption 2.1, we consider the nonlinear unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x), \tag{2.1}$$

which is a finite-sum minimization problem. For a very high dimension n and a very large dataset (with p instances), this challenging optimization problem is hard to solve.

For an arbitrary pair of vectors $a, b \in \mathbb{R}^n$, we write $a^\mathsf{T} b$ for their scalar product and we write ab^T for their dyadic product, i.e.

$$ab^\mathsf{T} := \begin{pmatrix} a_1b_1 & a_1b_2 & \cdots & a_1b_n \\ a_2b_1 & a_2b_2 & \cdots & a_2b_n \\ \vdots & \vdots & \ddots & \vdots \\ a_nb_1 & a_nb_2 & \cdots & a_nb_n \end{pmatrix}.$$

For arbitrary real numbers $\lambda_1, \ldots, \lambda_n$, the notation $\operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ is used for the diagonal matrix with the diagonal elements $\lambda_1, \ldots, \lambda_n$.

In general, we work with the Euclidean norm $\|\cdot\|_2$ as a special norm in $\mathbb{R}^n$, and $|||\cdot|||$ denotes the spectral norm of a square matrix. The term $\mathbf{1}$ means a square matrix with entries 1.

3. The Algorithm

Before presenting the proposed algorithm, we need to clarify the background of the formulas used, which avoid the time-consuming calculation of an inverse matrix.

Proposition 3.1. *Let $u \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$ be arbitrary parameters, and let $x, y \in \mathbb{R}^n$ be arbitrarily chosen where x has nonzero components. For every $\lambda_1, \dots, \lambda_n \in \mathbb{R}\backslash\{0\}$ with*

$$\big(\operatorname{diag}(\lambda_1, \dots, \lambda_n) + \alpha u u^\mathsf{T}\big) x = y \tag{3.1}$$

and

$$1 + \alpha u^\mathsf{T} \operatorname{diag}(\lambda_1, \dots, \lambda_n)^{-1} u \neq 0, \tag{3.2}$$

it holds

$$\big(\operatorname{diag}(\lambda_1, \dots, \lambda_n) + \alpha u u^\mathsf{T}\big)^{-1} = \operatorname{diag}(\lambda_1, \dots, \lambda_n)^{-1} - \beta \left(\frac{u_1}{\lambda_1}, \dots, \frac{u_n}{\lambda_n}\right) \left(\frac{u_1}{\lambda_1}, \dots, \frac{u_n}{\lambda_n}\right)^\mathsf{T} \tag{3.3}$$

with

$$\beta := \frac{\alpha}{1 + \alpha \sum_{i=1}^n \frac{u_i^2}{\lambda_i}}.$$

Proof. Let $u \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$ be arbitrary parameters, and let $x, y \in \mathbb{R}^n$ be arbitrarily chosen where x has nonzero components. If $\lambda_1, \dots, \lambda_n \in \mathbb{R}\backslash\{0\}$ solve the equality (3.1), then

$$\operatorname{diag}(\lambda_1, \dots, \lambda_n) x = y - \alpha u u^\mathsf{T} x$$

implying

$$\lambda_k = \frac{1}{x_k}\big(y_k - \alpha u_k \, u^\mathsf{T} x\big) \text{ for all } k \in \{1, \dots, n\}.$$

Thus, $\lambda_1, \dots, \lambda_n$ are uniquely defined. To calculate the inverse on the left side of the equation (3.3), we use the known Sherman-Morrison formula, and then we get with the abbreviation $\Lambda := \operatorname{diag}(\lambda_1, \dots, \lambda_n)$

$$\big(\Lambda + \alpha u u^\mathsf{T}\big)^{-1} = \Lambda^{-1} - \underbrace{\frac{\alpha}{1 + \alpha u^\mathsf{T} \Lambda^{-1} u}}_{=:\beta} \Lambda^{-1} u u^\mathsf{T} \Lambda^{-1}. \tag{3.4}$$

With

$$u^\mathsf{T} \Lambda^{-1} u = \sum_{k=1}^n \frac{u_k^2}{\lambda_k}$$

we conclude

$$\beta = \frac{\alpha}{1 + \alpha \sum_{k=1}^n \frac{u_k^2}{\lambda_k}}.$$

In addition, we can write

$$\begin{aligned} \Lambda^{-1} u u^\mathsf{T} \Lambda^{-1} &= \begin{pmatrix} \frac{1}{\lambda_1} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\lambda_n} \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} (u_1, \dots, u_n) \begin{pmatrix} \frac{1}{\lambda_1} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\lambda_n} \end{pmatrix} \\ &= \left(\frac{u_1}{\lambda_1}, \dots, \frac{u_n}{\lambda_n}\right) \left(\frac{u_1}{\lambda_1}, \dots, \frac{u_n}{\lambda_n}\right)^\mathsf{T}. \end{aligned}$$

Consequently, equation (3.3) follows from equation (3.4). □

Note that assumption (3.2) is easily satisfied if parameter $\alpha \in \mathbb{R}$ is chosen to be sufficiently close to zero. Or the components of the vector $u \in \mathbb{R}^n$ are chosen to be sufficiently close to zero.

Early work on matrices of the form $\operatorname{diag}(\lambda_1,\ldots,\lambda_n)+uu^{\mathsf{T}}$ (i.e., with $\alpha = 1$ in (3.1)) can already be found in [23] under the name "diagonal+rank-1". Matrices of the form "identity minus rank one" were examined in [24]. And matrices of the form "diagonal $\pm$ rank-1" were also studied in [5].

Remark 3.1. Under appropriate assumptions, the previous proposition with the equation (3.3) gives a simple formula for calculating the inverse of the symmetric matrix

$$\operatorname{diag}(\lambda_1,\ldots,\lambda_n)+\alpha uu^{\mathsf{T}} = \begin{pmatrix} \lambda_1+\alpha u_1^2 & \ldots & \alpha u_1 u_n \\ \vdots & & \vdots \\ \alpha u_n u_1 & \ldots & \lambda_n+\alpha u_n^2 \end{pmatrix}. \tag{3.5}$$

Note that the dyadic product is a matrix of rank 1 only. Nevertheless, the matrix in equation (3.5) can be used as a simplified Hessian matrix in a quasi-Newton method.

A simplification of the formula (3.3) can be obtained for the special case $u = (1,\ldots,1)^{\mathsf{T}}$.

Corollary 3.1. *Let the assumptions of Proposition 3.1 be fulfilled for $u = (1,\ldots,1)^{\mathsf{T}}$. Then we conclude*

$$\begin{pmatrix} \lambda_1+\alpha & \ldots & \alpha \\ \vdots & & \vdots \\ \alpha & \ldots & \lambda_n+\alpha \end{pmatrix}^{-1} = \begin{pmatrix} \frac{1}{\lambda_1}-\frac{\beta}{\lambda_1^2} & \cdots & -\frac{\beta}{\lambda_1\lambda_n} \\ \vdots & & \vdots \\ -\frac{\beta}{\lambda_n\lambda_1} & \cdots & \frac{1}{\lambda_n}-\frac{\beta}{\lambda_n^2} \end{pmatrix} \tag{3.6}$$

with

$$\beta := \frac{\alpha}{1+\alpha\sum_{k=1}^n \frac{1}{\lambda_k}}.$$

Proof. For the special case $u = (1,\ldots,1)^{\mathsf{T}}$, we get the dyadic product

$$\left(\frac{u_1}{\lambda_1},\ldots,\frac{u_n}{\lambda_n}\right)\left(\frac{u_1}{\lambda_1},\ldots,\frac{u_n}{\lambda_n}\right)^{\mathsf{T}} = \begin{pmatrix} \frac{1}{\lambda_1^2} & \cdots & \frac{1}{\lambda_1\lambda_n} \\ \vdots & & \vdots \\ \frac{1}{\lambda_n\lambda_1} & \cdots & \frac{1}{\lambda_n^2} \end{pmatrix},$$

and equality (3.6) follows from equality (3.3). □

The advantage of equality (3.6) is that one can compute the inverse in a very simple way. This is crucial for improved gradient methods in deep learning.

Remark 3.2. For an iteration index $j \in \mathbb{N}$, let $y^{j-1}, y^j \in \mathbb{R}^n$ be two iteration points of a gradient method and let the assumptions of Corollary 3.1 be satisfied for some parameter $\alpha \in \mathbb{R}$. If we set $x := y^j - y^{j-1}$ and $y := \nabla\hat{f}(y^j) - \nabla\hat{f}(y^{j-1})$ (where $\hat{f} : \mathbb{R}^n \to \mathbb{R}$ denotes any objective function of a minimization problem), the equality (3.1) can be written as

$$\begin{pmatrix} \lambda_1+\alpha & \ldots & \alpha \\ \vdots & & \vdots \\ \alpha & \ldots & \lambda_n+\alpha \end{pmatrix}(y^j - y^{j-1}) = \nabla\hat{f}(y^j) - \nabla\hat{f}(y^{j-1}).$$

This equality is a quasi-Newton equation, and from this we immediately get

$$\lambda_k = \frac{\frac{\partial \hat{f}}{\partial x_k}(y^j) - \frac{\partial \hat{f}}{\partial x_k}(y^{j-1}) - \eta}{y_k^j - y_k^{j-1}} \text{ for all } k \in \{1,\dots,n\}$$

with

$$\eta := \alpha \sum_{k=1}^{n} (y_k^j - y_k^{j-1}).$$

The λ_k can be easily calculated even for very large $n \in \mathbb{N}$. Next, a quasi-Newton direction d^j can be computed using the formula (3.6). So with the equality (3.6) we obtain

$$\begin{aligned} d^j &:= -\begin{pmatrix} \lambda_1+\alpha & \dots & \alpha \\ \vdots & & \vdots \\ \alpha & \dots & \lambda_n+\alpha \end{pmatrix}^{-1} \nabla \hat{f}(y^j) \\ &= -\begin{pmatrix} \frac{1}{\lambda_1} - \frac{\beta}{\lambda_1^2} & \dots & -\frac{\beta}{\lambda_1 \lambda_n} \\ \vdots & & \vdots \\ -\frac{\beta}{\lambda_n \lambda_1} & \dots & \frac{1}{\lambda_n} - \frac{\beta}{\lambda_n^2} \end{pmatrix} \nabla \hat{f}(y^j). \end{aligned}$$

Its k-th component can then simply be written as

$$d_k^j = \frac{-\frac{\partial \hat{f}}{\partial x_k}(y^j) + \delta}{\lambda_k} \text{ for all } k \in \{1,\dots,n\}$$

with

$$\delta := \beta \sum_{\ell=1}^{n} \frac{\frac{\partial \hat{f}}{\partial x_\ell}(y^j)}{\lambda_\ell}.$$

Finally, the next iteration point can simply be given as $y^{j+1} := y^j + d^j$. Note that no complicated matrix-vector products need to be computed to determine the iteration direction d^j, which is essential for computation on GPUs.

The formulas in the previous remark are now used to formulate a new mini-batch stochastic quasi-Newton method, where we replace the objective function $\hat{f}$ with the objective function used in the current mini-batch iteration. That is,

$$\hat{f}(x) := f_{B_i}(x) := \frac{1}{b} \sum_{k \in B_i} f_k(x) \text{ for all } x \in \mathbb{R}^n,$$

where $b \in \{1,\dots,p\}$ denotes the size of the current mini-batch, and the index set $B_i :\subseteq \{1,\dots,p\}$ (where i is the iteration index of the mini-batches) is sampled uniformly among index sets of size b.

Algorithm 3.1. (mini-batch stochastic quasi-Newton method)

Input: functions $f_1,\dots,f_p : \mathbb{R}^n \to \mathbb{R}$, starting vector $x^0 \in \mathbb{R}^n$,
uniform mini-batch size $b \in \{1,\dots,p\}$, quasi-Newton parameter $\alpha \in \mathbb{R}$,
maximal number of iterations $i_{\max} \geq 1$,
maximal iteration number $j_i \geq 1$ per i-th mini-batch ($i \in \{0,\dots,i_{\max}-1\}$),

5 uniform step size $\gamma_i > 0$ per i-th mini-batch ($i \in \{0, \ldots, i_{\max} - 1\}$).
6
7 $i := 0$ % initialization of index i
8 **while** $i \leq i_{\max} - 1$ **do**
9 $B_i :\subseteq \{1, \ldots, p\}$ % sampled uniformly among sets of size b
10 $j := 0$ % initialization of index j
11 $y^0 := x^i$ % initialization of iteration point
12 **while** $j \leq j_i - 1$ **do** % computations with a fixed mini-batch
13 **if** $j = 0$ **then**
14 $d^j := -\frac{1}{b} \sum_{k \in B_i} \nabla f_k(y^j) \qquad \left(= -\nabla f_{B_i}(y^j) \right)$ % iteration direction
15 **else**
16 $\eta := \alpha \sum_{k=1}^{n} (y_k^j - y_k^{j-1})$
17 **for** $k = 1 : 1 : n$ **do**
18 $\lambda_k := \dfrac{\frac{\partial f_{B_i}}{\partial x_k}(y^j) - \frac{\partial f_{B_i}}{\partial x_k}(y^{j-1}) - \eta}{y_k^j - y_k^{j-1}}$
19 **end for**
20 $\delta := \dfrac{\alpha}{1 + \alpha \sum_{k=1}^{n} \frac{1}{\lambda_k}} \sum_{k=1}^{n} \dfrac{\frac{\partial f_{B_i}}{\partial x_k}(y^j)}{\lambda_k}$
21 **for** $k = 1 : 1 : n$ **do**
22 $d_k^j = \dfrac{-\frac{\partial f_{B_i}}{\partial x_k}(y^j) + \delta}{\lambda_k}$ % components of the iteration direction
23 **end for**
24 **end if**
25 $y^{j+1} := y^j + \gamma_i d^j$ % new iteration point
26 $j := j + 1$
27 **end while**
28 $x^{i+1} := y^{j_i+1}$ % new iteration point of the outer loop
29 $i := i + 1$
30 **end while**
31
32 *Output:* $x^1, x^2, \ldots$.

Remark 3.3. (explanation of Algorithm 3.1)

(a) The so-called quasi-Newton parameter α in line 2 describes the parameter used in the quasi-Newton approximation

$$\begin{pmatrix} \lambda_1 + \alpha & \ldots & \alpha \\ \vdots & & \vdots \\ \alpha & \ldots & \lambda_n + \alpha \end{pmatrix}$$

of the Hessian matrix. If we choose $\alpha := 0$, we get $\eta = \delta = 0$. In this special case the iteration direction d^j in line 22 reduces to that of Algorithm 6.3 in [25]. In this sense, Algorithm 3.1 extends the quasi-Newton variant in [25, Alg. 6.3].

(b) With a chosen mini-batch in line 9, we do not perform only one iteration step, but a series of iteration steps - maximally j_i steps. This improves the convergence property of this method within a given mini-batch.

(c) The computations in lines 16, 18, 20, and 22 can be done simply - no complicated matrix-vector multiplications are required. These formulas are evaluated in Corollary 3.1 and Remark 3.2.

(d) Note that a step size (or learning rate) is used to calculate a new iteration point in line 25. This is necessary for the convergence of the stochastic method (reduction of the influence of gradient noise) under appropriate assumptions. See Corollary 4.2 for results with special step sizes.

(e) When implementing the algorithm, ensure that no divisions by zero occur.

The method in Algorithm 3.1 is motivated by a special quasi-Newton approach. However, the following remark provides an additional motivation.

Remark 3.4. (further motivation of the inner loop of Algorithm 3.1)
If we choose the special quasi-Newton parameter $\alpha = 0$, then the iteration direction d^j (line 22 of Algorithm 3.1) is simply the negative gradient $-\nabla f_{B_i}(y^j)$ where each component is multiplied by a number $\frac{1}{\lambda_k}$ (with $k \in \{1,\dots,n\}$). Rather than selecting a single step size, we now have a step size for each component of the negative gradient. This approach opens up much better possibilities, particularly when the optimization problem has a very large number of variables (compare [25, Alg. 6.3] for a batch gradient method).
Recall that Barzilai and Borwein [4] provided two formulas for a single step size based on a quasi-Newton equation. However, we work with multiple step sizes that are also derived from a quasi-Newton equation. Therefore, Algorithm 3.1 is an extension of the Barzilai-Borwein approach.
Note that we are not limited to selecting $\alpha = 0$. This suggests that the quasi-Newton parameter may have an additional value for this method.

In general, it is not guaranteed that the iteration direction d^j computed in line 22 of algorithm 3.1 is actually a descent direction. To implement this algorithm, the following algorithmic snippet can be added between lines 24 and 25.

Algorithmic snippet 3.2. (extension to Algorithm 3.1)

24.1 **if** $\left(\frac{1}{b}\sum_{k\in B_i}\nabla f_k(y^j)\right)^{\mathsf{T}} d^j > 0$ **then**
24.2 $d^j := -d^j$ % modified iteration direction
24.3 **end if**

Remark 3.5. The inequality in line 24.1 of the algorithmic snippet 3.2 says that the computed iteration direction d^j is an ascent direction for the mini-batch subproblem. To get a descent

direction, the sign is changed in line 24.2 for this direction. This extension to Algorithm 3.1 can be useful for highly nonlinear problems.

4. Convergence Results

For convergence investigations we distinguish between the inner while loop iteration and the outer while loop iteration in Algorithm 3.1. For the inner while loop iteration starting in line 12 of Algorithm 3.1, we assume that a mini-batch is fixed, and we begin with these investigations.

4.1. Superlinear Convergence for a Fixed Mini-Batch.

First, we recall some known notions and results.

Definition 4.1 (compare [26, Def. 7.1, (b)])**.** A sequence $(y^j)_{j\in\mathbb{N}}$ in $\mathbb{R}^n$ is called to converge *superlinearly* to some $\bar{y}\in\mathbb{R}^n$ iff there exists some sequence $(\varepsilon_j)_{j\in\mathbb{N}}$ of non-negative real numbers converging to zero with

$$\|y^{j+1}-\bar{y}\|_2 \le \varepsilon_j \|y^j-\bar{y}\|_2 \text{ for all } j\in\mathbb{N}.$$

Superlinear convergence is sometimes also called *Q-superlinear convergence*. Using the little-o notation, superlinear convergence of a sequence $(y^j)_{j\in\mathbb{N}}$ to some $\bar{y}$ means

$$\|y^{j+1}-\bar{y}\|_2 = o(\|y^j-\bar{y}\|_2) \text{ for all } j\in\mathbb{N}.$$

Next, we consider a fixed mini-batch given by an index set $B_i\subseteq\{1,\dots,p\}$ for a batch size $b\in\{1,\dots,p\}$. For this mini-batch, the objective function $f_{B_i}:\mathbb{R}^n\to\mathbb{R}$ is defined by

$$f_{B_i}(x) := \frac{1}{b}\sum_{k\in B_i} f_k(x) \text{ for all } x\in\mathbb{R}^n.$$

Theorem 4.1 ([27, Thm. 2.2], [28, Thm. 5.6], and [26, Korollar 7.9])**.** *Let the objective function $f_{B_i}:\mathbb{R}^n\to\mathbb{R}$ be twice continuously differentiable, let $(H_j)_{j\in\mathbb{N}}$ be a sequence of regular (n,n) matrices, let $y^0\in\mathbb{R}^n$ be a starting point and let a sequence $(y^j)_{j\in\mathbb{N}}$ be given by the iteration formular*

$$y^{j+1} := y^j - H_j^{-1}\nabla f_{B_i}(y^j) \text{ for all } j\in\mathbb{N}\cup\{0\} \tag{4.1}$$

with limit $\bar{y}\in\mathbb{R}^n$, where $y^j\neq\bar{y}$ for all $j\in\mathbb{N}$, and regular Hessian matrix $\nabla^2 f_{B_i}(\bar{y})$. Then the following statements are equivalent:

(a) The sequence $(y^j)_{j\in\mathbb{N}}$ converges superlinearly to $\bar{y}$ and $\nabla f_{B_i}(\bar{y})=0_n$.
(b) $\|(\nabla^2 f_{B_i}(y^j)-H_j)(y^{j+1}-y^j)\|_2 = o(\|y^{j+1}-y^j\|_2)$.
(c) $\|(\nabla^2 f_{B_i}(\bar{y})-H_j)(y^{j+1}-y^j)\|_2 = o(\|y^{j+1}-y^j\|_2)$.

This result was originally given by Dennis and Moré [27] and extended to the infinite dimensional case by Gruver and Sachs [28]. The formulation of Theorem 4.1 is taken from Geiger and Kanzow [26].

Now we apply Theorem 4.1 to a fixed mini-batch part of Algorithm 3.1.

Corollary 4.1. *Consider Algorithm 3.1 with starting point $x^0\in\mathbb{R}^n$ and fixed parameters $b\in\{1,\dots,p\}$, $\alpha\in\mathbb{R}$ and uniform step size $\gamma_i := 1$. Let B_i be an arbitrary mini-batch, let y^0 be the starting point for this mini-batch and assume that infinitly many iterations per mini-batch B_i are possible, i.e. $j_i := \infty$. Let the objective function $f_{B_i}:\mathbb{R}^n\to\mathbb{R}$ be twice continuously differentiable, let a sequence $(y^j)_{j\in\mathbb{N}}$ be generated by Algorithm 3.1. Let this sequence converge*

to some $\bar{y} \in \mathbb{R}^n$ with $y^j \neq \bar{y}$ for all $j \in \mathbb{N}$. Assume there are nonnegative real numbers $\delta_1, \delta_2, \ldots$ with $\lim_{j\to\infty} \delta_j = 0$ so that for the spectral norm $|||\cdot|||$ either the inequality

$$|||\nabla^2 f_{B_i}(y^j) - \operatorname{diag}(\lambda_1^j, \ldots, \lambda_n^j) - \alpha \mathbf{1}||| \leq \delta_j \tag{4.2}$$

or the inequality

$$|||\nabla^2 f_{B_i}(\bar{y}) - \operatorname{diag}(\lambda_1^j, \ldots, \lambda_n^j) - \alpha \mathbf{1}||| \leq \delta_j \tag{4.3}$$

is fulfilled. Here the numbers $\lambda_1^j, \ldots, \lambda_n^j$ are the numbers determined in line 18 of Algorithm 3.1 at iteration j, and $\mathbf{1}$ denotes the (n,n) matrix with entries 1. Then the sequence $(y^j)_{j\in\mathbb{N}}$ convergences superlinearly to $\bar{y}$ and $\nabla f_{B_i}(\bar{y}) = 0_n$.

Proof. Let $y^0 \in \mathbb{R}^n$ be the starting point for the mini-batch B_i, and let $(y^j)_{j\in\mathbb{N}}$ be an infinite sequence generated by Algorithm 3.1 with limit $\bar{y} \in \mathbb{R}^n$. In this algorithm we work with the iteration formula

$$\begin{aligned} y^{j+1} &= y^j + \underbrace{\gamma_i}_{=1} d^j \text{ (see line 25 in Alg. 3.1)} \\ &= y^j - \left(\operatorname{diag}(\lambda_1^j, \ldots, \lambda_n^j) + \alpha \mathbf{1}\right)^{-1} \nabla f_{B_i}(y^j) \text{ for all } j \in \mathbb{N} \cup \{0\} \\ &\qquad \text{(see Remark 3.2 and line 22 and the lines before in Alg. 3.1).} \end{aligned}$$

This iteration formula is of the form of the one given in (4.1) for

$$H_j^{-1} := \left(\operatorname{diag}(\lambda_1^j, \ldots, \lambda_n^j) + \alpha \mathbf{1}\right)^{-1} \text{ for all } j \in \mathbb{N} \cup \{0\}.$$

If the inequality (4.2) is satisfied, then we get

$$\begin{aligned} &\|(\nabla^2 f_{B_i}(y^j) - H_j)(y^{j+1} - y^j)\|_2 \\ &\quad \leq |||\nabla^2 f_{B_i}(y^j) - H_j|||\, \|y^{j+1} - y^j\|_2 \\ &\quad = |||\nabla^2 f_{B_i}(y^j) - \operatorname{diag}(\lambda_1^j, \ldots, \lambda_n^j) - \alpha \mathbf{1}|||\, \|y^{j+1} - y^j\|_2 \\ &\quad \leq \delta_j \|y^{j+1} - y^j\|_2 \\ &\quad = o(\|y^{j+1} - y^j\|_2) \text{ for all } j \in \mathbb{N} \cup \{0\}. \end{aligned}$$

With the same arguments as before we conclude under the assumption (4.3)

$$\|(\nabla^2 f_{B_i}(\bar{y}) - H_j)(y^{j+1} - y^j)\|_2 \leq o(\|y^{j+1} - y^j\|_2) \text{ for all } j \in \mathbb{N} \cup \{0\}.$$

Thus, by Theorem 4.1, for the two assumptions (4.2) or (4.3), the infinite sequence $(y^j)_{j\in\mathbb{N}}$ converges superlinearly to $\bar{y}$ and $\nabla f_{B_i}(\bar{y}) = 0_n$. □

Remark 4.1.

(a) The assumption (4.3) in Corollary 4.1 requires that the Hessian of the objective function has a special structure at the limit $\bar{y}$. Here, Hessians with equal entries outside the diagonal or, as a special case, a diagonal Hessian matrix (as in [25, Alg. 6.3]) are preferred. A Hessian with an arbitrarily symmetric structure would require a standard quasi-Newton method with the disadvantage of drastically high computation times on GPUs.

(b) In Corollary 4.1 the step size is set to one, i.e. we actually do not need a step size for this quasi-Newton method. It is pointed out in [26, Aufgabe 7.4] (see also [27, Cor. 2.3]) that in the case of a general step size, parts (b) and (c) of Theorem 4.1 must be extended

by the requirement that these step sizes converge to 1 (and not to 0). A step size is used in Algorithm 3.1 because it is crucial for a mini-batch method.

(c) In practice, we do not consider infinitly many iteration points as in Corollary 4.1 but only a limited number of iterations for the mini-batch subproblem.

4.2. **Convergence Analysis of the Outer Loop.**
For convergence estimates of Algorithm 3.1, one must consider the entire algorithmic approach. It certainly makes sense to do several iterations per mini-batch, such as j_i iterations in Algorithm 3.1. The essential iteration points of this method are $x^1, x^2, \ldots, x^{i_{\max}}$ starting from x^0. Below are convergence investigations for these iteration points.

But first we list some known definitions and properties. Recall that a function $f : \mathbb{R}^n \to \mathbb{R}$ is called μ-*strongly convex* for some $\mu > 0$ iff for all $x, y \in \mathbb{R}^n$ we have

$$\mu \frac{\lambda(1-\lambda)}{2} \|x-y\|_2^2 + f\big(\lambda x + (1-\lambda)y\big) \leq \lambda f(x) + (1-\lambda) f(y) \text{ for all } \lambda \in [0,1]$$

(for instance, compare [25, Def. 1.25]). Note that every continuous and μ-strongly convex function $f : \mathbb{R}^n \to \mathbb{R}$ (with $\mu > 0$) has a unique minimal point (see, for example, [29, Lemma 2.13]). And for every differentiable and μ-strongly convex function $f : \mathbb{R}^n \to \mathbb{R}$ (with $\mu > 0$) it holds

$$f(y) \geq f(x) + \nabla f(x)^{\mathsf{T}}(y-x) + \frac{\mu}{2}\|y-x\|_2^2 \text{ for all } x, y \in \mathbb{R}^n \tag{4.4}$$

(see [29, Lemma 2.14]).

While the iteration points in the inner while loop of Algorithm 3.1 are obtained deterministically, the investigation of the iteration sequence in the outer while loop of this method requires a stochastic approach. For an arbitrary iteration index $i \in \{0, \ldots, i_{\max}\}$, an index set $B_i :\subseteq \{1, \ldots, p\}$ is chosen that is uniformly sampled among index sets of size b. Then we use the definition $f_{B_i} : \mathbb{R}^n \to \mathbb{R}$ with

$$f_{B_i}(x) := \frac{1}{b} \sum_{k \in B_i} f_k(x) \text{ for all } x \in \mathbb{R}^n$$

and we obtain

$$\nabla f_{B_i}(x) := \frac{1}{b} \sum_{k \in B_i} \nabla f_k(x) \text{ for all } x \in \mathbb{R}^n$$

for differentiable functions f_k with $k \in B_i$. Such a mini-batch is sampled with probability $\frac{1}{\binom{p}{b}}$ and the expectation of the mini-batch gradient can be written as

$$E_b(\nabla f_{B_i}(x)) = \frac{1}{\binom{p}{b}} \sum_{k \in B_i} \nabla f_k(x) \text{ for all } x \in \mathbb{R}^n,$$

which is equal to $\nabla f(x)$ (see [29, Def. 6.2]). If the optimization problem (2.1) has a unique minimal solution $\bar{x} \in \mathbb{R}^n$, then the so-called *mini-batch gradient noise* is defined by

$$\bar{\sigma}_b := V_b(\nabla f_{B_i}(\bar{x}))$$

where $V_b(\nabla f_{B_i}(\bar{x}))$ denotes the variance of the mini-batch gradient at the minimal solution $\bar{x}$ (see [29, Def. 6.3]). For the treatment of gradient noise, we recall another important notion.

The objective function f is said to be *L_b-smooth in expectation* (for some $L_b > 0$) iff

$$\frac{1}{2L_b} E_b(\|\nabla f_{B_i}(y) - \nabla f_{B_i}(x)\|_2^2) \leq f(y) - f(x) - \nabla f(x)^\mathsf{T}(y-x) \text{ for all } x, y \in \mathbb{R}^n$$

(compare [29, Definition 6.4]). For convergence proofs, the following variance transfer lemma plays an important role.

Lemma 4.1 (Lemma 6.7 by Garrigos and Gower [29]). *Let Assumption 2.1 be satisfied, let the assumed setting of this subsection be given, and let every function f_k (with $k \in \{1, \ldots, p\}$) be differentiable, convex and L_k-smooth, i.e. ∇f_k is Lipschitz continuous with constant $L_k > 0$. Then there is some $L_b > 0$ so that*

$$E_b(\|\nabla f_{B_i}(x)\|_2^2) \leq 4L_b(f(x) - f(\bar{x})) + 2\bar{\sigma}_b \text{ for all } x \in \mathbb{R}^n.$$

The following theorem extends a result of Garrigos and Gower [29, Thm. 6.11] to the case of a mini-batch method with several quasi-Newton steps within each mini-batch.

Theorem 4.2. *Consider the optimization problem (2.1) under Assumption 2.1. Let every function f_k (with $k \in \{1, \ldots, p\}$) be differentiable, convex and L_k-smooth (for some $L_k > 0$), and let the objective function f be μ-strongly convex for some $\mu > 0$. Let $\bar{x} \in \mathbb{R}^n$ denote the unique minimal solution of the optimization problem (2.1). Let $(x^i)_{i \in \{1, \ldots, i_{max}\}}$ be a sequence generated by Algorithm 3.1 with starting point $x^0 \in \mathbb{R}^n$ and step size γ_i with*

$$0 < \gamma_i \leq \frac{c}{2L_b m d} \text{ for all } i \in \{0, \ldots, i_{max} - 1\}.$$

The constant $L_b > 0$ results from Lemma 4.1, and the constants c, m, and d are given as follows. Let there exist some $c, d > 0$ such that for all $i \in \{0, \ldots, i_{max}\}$, the inequalities

$$(x^i - \bar{x})^\mathsf{T} \sum_{j=0}^{j_i} H_j^{-1} \nabla f_{B_i}(y^j) \geq c(x^i - \bar{x})^\mathsf{T} \nabla f_{B_i}(x^i) \tag{4.5}$$

with $H_j := \operatorname{diag}(\lambda_1^j, \ldots, \lambda_n^j) + \alpha \mathbf{1}$ for all $j \in \{0, \ldots, j_i\}$ and

$$\sum_{j=0}^{j_i} \|\nabla f_{B_i}(y^j)\|_2^2 \leq d \|\nabla f_{B_i}(x^i)\|_2^2 \tag{4.6}$$

are satisfied. Further, assume for the spectral norm $|||\cdot|||$ that there is some $m > 0$ such that for each mini-batch

$$|||H_j^{-1}|||^2 \leq m \text{ for all } j \in \{0, \ldots, j_i\}. \tag{4.7}$$

Then we get for all $i \in \{0, \ldots, i_{max} - 1\}$

$$E_b(\|x^{i+1} - \bar{x}\|_2^2) \leq (1 - c\mu\gamma_i) E_b(\|x^i - \bar{x}\|_2^2) + 2md\gamma_i^2 \bar{\sigma}_b. \tag{4.8}$$

Proof. Since the objective function f is continuous and μ-strongly convex, it is well-known that the optimization problem (2.1) has a unique minimal solution $\bar{x} \in \mathbb{R}^n$ (see [29, Lemma 2.13]). For an arbitrary $i \in \{0, \ldots, i_{\max} - 1\}$ we then get

$$\begin{aligned} \|x^{i+1} - \bar{x}\|_2^2 &= \|x^i - \bar{x} + x^{i+1} - x^i\|_2^2 \\ &= \|x^i - \bar{x}\|_2^2 + 2(x^i - \bar{x})^\mathsf{T}(x^{i+1} - x^i) + \|x^{i+1} - x^i\|_2^2. \end{aligned} \tag{4.9}$$

With the assumption (4.5) the term $(x^i-\bar{x})^{\mathsf{T}}(x^{i+1}-x^i)$ in (4.9) can be written as

$$\begin{aligned}(x^i-\bar{x})^{\mathsf{T}}(x^{i+1}-x^i) &= (x^i-\bar{x})^{\mathsf{T}}\gamma_i\sum_{j=0}^{j_i}d^j\\ &= -\gamma_i(x^i-\bar{x})^{\mathsf{T}}\sum_{j=0}^{j_i}H_j^{-1}\nabla f_{B_i}(y^j)\\ &\leq -c\gamma_i(x^i-\bar{x})^{\mathsf{T}}\nabla f_{B_i}(x^i). \end{aligned}\tag{4.10}$$

By the assumptions (4.7) and (4.6) we get for the last term in (4.9)

$$\begin{aligned}\|x^{i+1}-x^i\|_2^2 &= \Big\|-\gamma_i\sum_{j=0}^{j_i}H_j^{-1}\nabla f_{B_i}(y^j)\Big\|_2^2\\ &\leq \gamma_i^2\sum_{j=0}^{j_i}|||H_j^{-1}|||^2\cdot\|\nabla f_{B_i}(y^j)\|_2^2\\ &\leq m\gamma_i^2\sum_{j=0}^{j_i}\|\nabla f_{B_i}(y^j)\|_2^2\\ &\leq md\gamma_i^2\|\nabla f_{B_i}(x^i)\|_2^2. \end{aligned}\tag{4.11}$$

Thus the equation (4.9) together with the inequalities (4.10) and (4.11) implies

$$\|x^{i+1}-\bar{x}\|_2^2 \leq \|x^i-\bar{x}\|_2^2-2c\gamma_i(x^i-\bar{x})^{\mathsf{T}}\nabla f_{B_i}(x^i)+md\gamma_i^2\|\nabla f_{B_i}(x^i)\|_2^2.$$

With this inequality we can provide an upper bound for the conditional expectation $E_b(\|x^{i+1}-\bar{x}\|_2^2\mid x^i)$ and we get

$$\begin{aligned}E_b(\|x^{i+1}-\bar{x}\|_2^2\mid x^i) &\leq \|x^i-\bar{x}\|_2^2-2c\gamma_i(x^i-\bar{x})^{\mathsf{T}}E_b(\nabla f_{B_i}(x^i)\mid x^i)\\ &\quad +md\gamma_i^2E_b(\|\nabla f_{B_i}(x^i)\|_2^2\mid x^i). \end{aligned}\tag{4.12}$$

Since $\nabla f_{B_i}(x^i)$ is an unbiased estimator of $\nabla f(x^i)$ and f is μ-strongly convex (we use the inequality (4.4)), the inequality (4.12) leads to

$$\begin{aligned}E_b(\|x^{i+1}-\bar{x}\|_2^2\mid x^i) &\leq \|x^i-\bar{x}\|_2^2-2c\gamma_i\underbrace{(x^i-\bar{x})^{\mathsf{T}}\nabla f(x^i)}_{\geq f(x^i)-f(\bar{x})+\frac{\mu}{2}\|x^i-\bar{x}\|_2^2}\\ &\quad +md\gamma_i^2E_b(\|\nabla f_{B_i}(x^i)\|_2^2\mid x^i)\\ &\leq (1-c\mu\gamma_i)\|x^i-\bar{x}\|_2^2-2c\gamma_i(f(x^i)-f(\bar{x}))\\ &\quad +md\gamma_i^2E_b(\|\nabla f_{B_i}(x^i)\|_2^2\mid x^i). \end{aligned}\tag{4.13}$$

By Lemma 4.1 we obtain the following upper bound for the conditional expectation on the right side of the inequality (4.13)

$$E_b(\|\nabla f_{B_i}(x^i)\|_2^2\mid x^i) \leq 4L_b(f(x^i)-f(\bar{x}))+2\bar{\sigma}_b.$$

This upper bound is now used in the inequality (4.13), i.e.

$$\begin{aligned} E_b(\|x^{i+1}-\bar{x}\|_2^2 \mid x^i) &\leq (1-c\mu\gamma_i)\|x^i-\bar{x}\|_2^2 - 2c\gamma_i(f(x^i)-f(\bar{x})) \\ &\quad +4L_b md\gamma_i^2(f(x^i)-f(\bar{x}))+2md\gamma_i^2\bar{\sigma}_b \\ &= (1-c\mu\gamma_i)\|x^i-\bar{x}\|_2^2+2\gamma_i(2L_b md\gamma_i-c)(f(x^i)-f(\bar{x})) \\ &\quad +2md\gamma_i^2\bar{\sigma}_b. \end{aligned} \tag{4.14}$$

By assumption we have $\gamma_i \leq \frac{c}{2L_b md}$, i.e. $2L_b md\gamma_i - c \leq 0$, and since $f(x^i) \geq f(\bar{x})$, we conclude from the inequality (4.14)

$$E_b(\|x^{i+1}-\bar{x}\|_2^2 \mid x^i) \leq (1-c\mu\gamma_i)\|x^i-\bar{x}\|_2^2+2md\gamma_i^2\bar{\sigma}_b.$$

If we consider the expectation of the conditional expectation, then by the law of total expectation we obtain

$$E_b(\|x^{i+1}-\bar{x}\|_2^2) \leq (1-c\mu\gamma_i)\,E_b(\|x^i-\bar{x}\|_2^2)+2md\gamma_i^2\bar{\sigma}_b,$$

and the inequality (4.8) is shown. □

The inequality (4.8) in Theorem 4.1 suggests that the influence of gradient noise can be reduced by working with small step sizes $\gamma_i > 0$. The next corollary, which follows the lines of proof by Bottou, Curties, and Nocedal [8, Thm. 4.7] and extends their result to the mini-batch variant of this paper, gives a convergence result for a special step size rule.

Corollary 4.2. *Consider the optimization problem (2.1), and let the assumptions of Theorem 4.2 be satisfied. In Algorithm 3.1 choose an arbitrary starting point $x^0 \neq \bar{x}$, where $\bar{x}$ denotes the unique minimal solution of the optimization problem (2.1), and the special step sizes*

$$\gamma_i := \frac{\rho}{\tau+i} \text{ for all } i \in \{0,\dots,i_{max}-1\}$$

for real numbers ρ, τ with $\rho > \frac{1}{c\mu}$ (for the constants μ, c, m, and d see the assumptions in Theorem 4.2) and $\tau > 1$. Then for $\psi := \max\left\{\frac{2md\rho^2\bar{\sigma}_b}{c\mu\rho-1}, \tau\|x^0-\bar{x}\|_2^2\right\} > 0$ we have

$$E_b(\|x^i-\bar{x}\|_2^2) \leq \frac{\psi}{\tau+i} \text{ for all } i \in \{0,\dots,i_{max}\}. \tag{4.15}$$

Proof. We show the inequality (4.15) by induction on i. The base case for $i = 0$ gives

$$E_b(\|x^0-\bar{x}\|_2^2) = \|x^0-\bar{x}\|_2^2 = \frac{\tau\|x^0-\bar{x}\|_2^2}{\tau} \leq \frac{\psi}{\tau}.$$

For the induction step assume that the inequality (4.15) holds for an arbitrary $i \in \{0,\dots,i_{\max}-1\}$. Then we conclude with the inequality (4.8) in Theorem 4.2 together with the assumptions

of this corollary

$$\begin{aligned}
E_b(\|x^{i+1}-\bar{x}\|_2^2) &\leq \left(1-\frac{c\mu\rho}{\tau+i}\right)E_b(\|x^i-\bar{x}\|_2^2)+\frac{2md\rho^2}{(\tau+i)^2}\bar{\sigma}_b \\
&\leq \left(1-\frac{c\mu\rho}{\tau+i}\right)\frac{\psi}{\tau+i}+\frac{2md\rho^2}{(\tau+i)^2}\bar{\sigma}_b \\
&= \frac{\tau+i-1}{(\tau+i)^2}\psi\underbrace{-\frac{c\mu\rho-1}{(\tau+i)^2}\psi+\frac{2md\rho^2\bar{\sigma}_b}{(\tau+i)^2}}_{\leq 0 \text{ because } \psi\geq\frac{2md\rho^2\bar{\sigma}_b}{c\mu\rho-1}} \\
&\leq \frac{\tau+i-1}{(\tau+i)^2}\psi \\
&\leq \frac{\tau+i-1}{(\tau+i)^2-1}\psi \\
&= \frac{\tau+i-1}{(\tau+i+1)(\tau+i-1)}\psi \\
&= \frac{\psi}{\tau+i+1},
\end{aligned}$$

and the proof is complete. □

Corollary 4.2 shows that there is a decreasing upper bound on the expectation of $\|x^i-\bar{x}\|_2^2$ when we use a special step size rule. In this case, we can eliminate the term with the mini-batch gradient noise in the upper bound of the expectation of the distance between an iteration point and the minimal solution of the optimization problem. In this way, it makes sense to work with a sequence of decreasing step sizes. Another way to reduce the effect of noisy gradient estimates is to gradually increase the mini-batch sizes. In this paper, we work with a constant size b of a mini-batch to avoid a gradual increase of the computation time per mini-batch. We choose the mini-batch size b to be as large as possible under the assumption that reasonable computation times are achieved.

The convexity of the functions $f_1,\ldots,f_p$ is an essential assumption in Theorem 4.2 and Corollary 4.2. If this assumption is not given, some adaptations are necessary. The first thing to mention is the assumption of the Lipschitz continuity of the objective function f, which is implied by the Lipschitz continuity of all functions $f_1,\ldots,f_p$. Furthermore, a suitable noise level $\sigma>0$ of the gradient estimation is needed (for this term compare [11, inequality (2.2)]). This noise level is actually an upper bound, and it is used in our assumption (4.18) in the next theorem. Assumption (4.18) was already given in [11, inequality (2.6)].

The following theorem describes a certain descent property of the objective function value at each iteration point of Algorithm 3.1 without assuming convexity. The proof is based on the proofs given in [30, Lemma 2] and [11, Lemma 2.2], and extends them to Algorithm 3.1.

Theorem 4.3. *Consider the optimization problem (2.1) under Assumption 2.1. Let every function f_k (with $k\in\{1,\ldots,p\}$) be differentiable and let the gradient of the objective function f be Lipschitz continuous with Lipschitz constant $L>0$. i.e.*

$$\|\nabla f(u)-\nabla f(v)\|_2\leq L\|u-v\|_2 \text{ for all } u,v\in\mathbb{R}^n.$$

Let $(x^i)_{i\in\{1,\dots,i_{max}\}}$ be a sequence generated by Algorithm 3.1 with starting point $x^0 \in \mathbb{R}^n$ and step size γ_i with

$$0 < \gamma_i \leq \frac{c}{Ld} \text{ for all } i \in \{0,\dots,i_{max}-1\},$$

where the constants c and d are given as follows. Let there exist some $c,d>0$ such that for all $i \in \{0,\dots,i_{max}\}$, the inequalities

$$\nabla f(x^i)^\mathsf{T} \sum_{j=0}^{j_i} H_j^{-1} \nabla f_{B_i}(y^j) \geq c \nabla f(x^i)^\mathsf{T} \nabla f_{B_i}(x^i) \tag{4.16}$$

with $H_j := \operatorname{diag}(\lambda_1^j,\dots,\lambda_n^j) + \alpha \mathbf{1}$ for all $j \in \{0,\dots,j_i\}$ and

$$\Big\| \sum_{j=0}^{j_i} H_j^{-1} \nabla f_{B_i}(y^j) \Big\|_2^2 \leq d \|\nabla f_{B_i}(x^i)\|_2^2 \tag{4.17}$$

are satisfied. Further assume that there is some noise level $\sigma > 0$ of the gradient estimation with

$$E_b(\|\nabla f_{B_i}(x^i) - \nabla f(x^i)\|_2^2 \mid x^i) \leq \frac{\sigma^2}{b}. \tag{4.18}$$

Then we get for all $i \in \{0,\dots,i_{max}-1\}$

$$E_b(f(x^{i+1})|x^i) \leq f(x^i) - \frac{c}{2}\gamma_i \|\nabla f(x^i)\|_2^2 + \frac{Ld}{2b}\gamma_i^2 \sigma^2. \tag{4.19}$$

Proof. It is well-known (e.g., see [31, Lemma 4]) that Lipschitz continuity of the gradient ∇f implies

$$f(u) \leq f(v) + \nabla f(v)^\mathsf{T}(u-v) + \frac{L}{2}\|u-v\|_2^2 \text{ for all } u,v \in \mathbb{R}^n.$$

Using inequalities (4.16) and (4.17), we obtain for an arbitrary $i \in \{0,\dots,i_{\max}-1\}$

$$\begin{aligned} f(x^{i+1}) &\leq f(x^i) + \nabla f(x^i)^\mathsf{T}(x^{i+1}-x^i) + \frac{L}{2}\|x^{i+1}-x^i\|_2^2 \\ &= f(x^i) - \gamma_i \nabla f(x^i)^\mathsf{T} \sum_{j=0}^{j_i} H_j^{-1} \nabla f_{B_i}(y^j) + \frac{L}{2}\Big\| \sum_{j=0}^{j_i} H_j^{-1} \nabla f_{B_i}(y^j) \Big\|_2^2 \\ &\leq f(x^i) - c\gamma_i \nabla f(x^i)^\mathsf{T} \nabla f_{B_i}(x^i) + \frac{L}{2}\gamma_i^2 d \|\nabla f_{B_i}(x^i)\|_2^2. \end{aligned}$$

From this inequality, we can derive an upper bound of the conditional expectation

$$\begin{aligned} E_b(f(x^{i+1}) \mid x^i) &\leq f(x^i) - c\gamma_i \nabla f(x^i)^\mathsf{T} \underbrace{E_b(\nabla f_{B_i}(x^i) \mid x^i)}_{=\nabla f(x^i)} + \frac{L}{2}\gamma_i^2 d\, E_b(\|\nabla f_{B_i}(x^i)\|_2^2 \mid x^i) \\ &= f(x^i) - c\gamma_i \|\nabla f(x^i)\|_2^2 + \frac{L}{2}\gamma_i^2 d\, E_b(\|\nabla f_{B_i}(x^i)\|_2^2 \mid x^i). \end{aligned} \tag{4.20}$$

Next, notice

$$\begin{aligned} & E_b(\|\nabla f_{B_i}(x^i)\|_2^2 \mid x^i) \\ &= E_b(\|\nabla f(x^i) + \nabla f_{B_i}(x^i) - \nabla f(x^i)\|_2^2 \mid x^i) \\ &= \|\nabla f(x^i)\|_2^2 + 2\underbrace{E_b(\nabla f(x^i)^\mathsf{T}(\nabla f_{B_i}(x^i) - \nabla f(x^i)) \mid x^i)}_{=0 \text{ because } E_b(\nabla f_{B_i}(x^i) \mid x^i) = \nabla f(x^i)} + \underbrace{E_b(\|\nabla f_{B_i}(x^i) - \nabla f(x^i)\|_2^2 \mid x^i)}_{\leq \frac{\sigma^2}{b} \text{ by assumption (4.18)}} \\ &\leq \|\nabla f(x^i)\|_2^2 + \frac{\sigma^2}{b}. \end{aligned} \tag{4.21}$$

With inequality (4.21), inequality (4.20) can then be written as

$$\begin{aligned} E_b(f(x^{i+1}) \mid x^i) &\leq f(x^i) - \underbrace{\left(c - \frac{L}{2}\gamma_i d\right)}_{\geq \frac{c}{2} \text{ because } \gamma_i \leq \frac{c}{Ld}} \gamma_i \|\nabla f(x^i)\|_2^2 + \frac{Ld}{2b}\gamma_i^2\sigma^2 \\ &\leq f(x^i) - \frac{c}{2}\gamma_i\|\nabla f(x^i)\|_2^2 + \frac{Ld}{2b}\gamma_i^2\sigma^2, \end{aligned}$$

and the assertion (4.19) is shown. □

As stated in Theorem 4.3, the conditional expectation of the objective function value at a new iteration point may be smaller than the value at the previous iteration point if the noise level is nearly zero. For a large noise level, a real decrease in objective function values may be improbable.

5. Numerical Results

This section focuses on large-scale optimization problems involving between 1 million and 200/400 million variables. The reported results demonstrate the potential benefits for deep learning. We start with an illustrative example of a scalable optimization problem that may have a large number of variables.

Example 5.1. For an arbitrary $n \in \mathbb{N}$, we consider the unconstrained optimization problem $\min_{x \in \mathbb{R}^n} f(x)$ with the objective function $f : \mathbb{R}^n \to \mathbb{R}$ given by

$$f(x) := \frac{1}{n}\sum_{k=1}^{n} \underbrace{\left(2\left(e^{x_1^2 + x_k^2} - 1\right) + 5\left(\sum_{\ell=1}^{n} x_\ell\right)^2\right)}_{=: f_k(x)} \text{ for all } x = (x_1, \ldots, x_n) \in \mathbb{R}^n$$

where the functions $f_k : \mathbb{R}^n \to \mathbb{R}$ (with $k \in \{1, \ldots, n\}$) are given by

$$f_k(x) := 2\left(e^{x_1^2 + x_k^2} - 1\right) + 5\left(\sum_{\ell=1}^{n} x_\ell\right)^2 \text{ for all } x = (x_1, \ldots, x_n) \in \mathbb{R}^n. \tag{5.1}$$

In this particular case we have $p := n$. The second summand on the right side of the equation (5.1) is a special quadratic form, which can be written as

$$5\left(\sum_{\ell=1}^{n} x_\ell\right)^2 = x^\mathsf{T} \begin{pmatrix} 5 & \ldots & 5 \\ \vdots & \ddots & \vdots \\ 5 & \ldots & 5 \end{pmatrix} x,$$

and its Hessian matrix equals $\begin{pmatrix} 10 & \dots & 10 \\ \vdots & \ddots & \vdots \\ 10 & \dots & 10 \end{pmatrix}$. Since this matrix has equal entries, and by neglecting the Hessian of the first summand on the right side of the equality (5.1), it makes sense to choose the quasi-Newton parameter $\alpha := 10$ in line 2 of Algorithm 3.1. Next, we determine the gradient of the functions f_k. For $k = 1$ we have

$$\nabla f_1(x) = 8e^{2x_1^2} \begin{pmatrix} x_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} 10 & \dots & 10 \\ \vdots & \ddots & \vdots \\ 10 & \dots & 10 \end{pmatrix} x \text{ for all } x = (x_1, \dots, x_n) \in \mathbb{R}^n,$$

and for arbitrary $k \in \{2, \dots, n\}$ we get

$$\nabla f_k(x) = 4e^{x_1^2 + x_k^2} \begin{pmatrix} x_1 \\ 0 \\ \vdots \\ 0 \\ x_k \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} 10 & \dots & 10 \\ \vdots & \ddots & \vdots \\ 10 & \dots & 10 \end{pmatrix} x \text{ for all } x = (x_1, \dots, x_n) \in \mathbb{R}^n$$

where x_k is the k-th component of the first vector on the right side of this equality. These gradients are used in Algorithm 3.1. Since the objective function f is nonnegative and $f(0_n) = 0$, the vector 0_n is a global minimal solution of the considered optimization problem and therefore we have $\nabla f(0_n) = 0_n$. Hence, it makes sense to choose starting points in a neighborhood of the origin. Another reason for this choice is that outside such a neighborhood of 0_n and for a very large dimension n, the function values of f can increase sharply. Therefore, we choose a scalable starting point, explicitly given by $x^0 := \frac{100}{n}(1, \dots, 1)^\mathsf{T}$ (note that for very large n, we already have $f(x^0) \approx 50,000$).

First, consider this example with the dimension $n := 1{,}000{,}000$. The mini-batch size is chosen as $b := 100{,}000$, the maximal number of iterations per i-th mini-batch is chosen as $j_i := 10$, and we use a constant step size $\gamma_i := 10^{-7}$. The computations are performed with MATLAB R2024b on an 8-core iMac. The numerical results obtained with Algorithm 3.1 are illustrated in Figure 1. The iteration point after the fifth iteration is $x^5 := (-4 \cdot 10^{-11}, 7.9999 \cdot 10^{-17}, 7.9999 \cdot 10^{-17}, \dots)^\mathsf{T}$ and the CPU time required is 3.48 seconds. However, only two iterations are actually needed (see Figure 1).

Next, Table 1 shows numerical results for different dimensions of this example, starting with the case of one million variables already covered. Unless otherwise noted, the parameters are chosen as previously specified. Due to high CPU times for $n = 400{,}000{,}000$, the method was stopped after 5 iterations, although convergence was not reached. The algorithmic performance depends strongly on a good choice of parameters, where the mini-batch size b and the constant step size γ_i play an important role. Because of the quasi-Newton part of Algorithm 3.1 and its

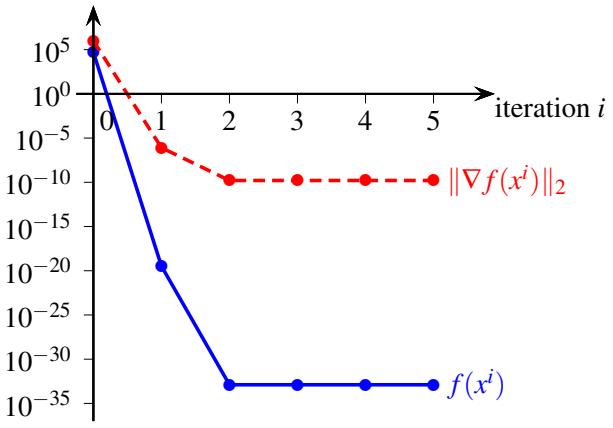

FIGURE 1. Semilogarithmic illustration of the function values $f(x^i)$ and the norms $\|\nabla f(x^i)\|_2$ at the iteration points x^i including the starting point (with $i \in \{0,\ldots,5\}$) in Example 5.1 for the dimension $n = 1{,}000{,}000$.

number n of variables	mini-batch size b	constant step size γ_i	max. iter. no. $i_{\max}$	$f(x_{i_{\max}})$	$\|\nabla f(x_{i_{\max}})\|_2$	CPU time in seconds
1,000,000	100,000	10^{-7}	5	$1.28 \cdot 10^{-33}$	$1.6 \cdot 10^{-10}$	3.48
10,000,000	1,000,000	10^{-8}	5	$1.28 \cdot 10^{-39}$	$1.6 \cdot 10^{-12}$	35.33
100,000,000	1,000,000	10^{-9}	5	$1.28 \cdot 10^{-45}$	$1.6 \cdot 10^{-14}$	363.59
200,000,000	1,000,000	$5 \cdot 10^{-10}$	5	$2 \cdot 10^{-47}$	$4 \cdot 10^{-15}$	847.38
400,000,000	1,000,000	10^{-10}	5	$3.0233 \cdot 10^{2}$	$1.555 \cdot 10^{6}$	3,612.84

TABLE 1. Numerical results for Algorithm 3.1 in Example 5.1 using MATLAB R2024b on an 8-core iMac.

good convergence behavior, one can work with relatively large mini-batch sizes in this example, which seems to be characteristic for this method.

The optimization problem in Example 5.1 fits the theoretical approach of this paper because the Hessian of the functions f_k have almost equal non-diagonal entries. Therefore, we now discuss an optimization problem for which this property is not given.

Example 5.2. Let the dimension $n \in \mathbb{N}$ be an arbitrary even number and let $p := n$. The functions $f_k : \mathbb{R}^n \to \mathbb{R}$ (with $k \in \{1,\ldots,n\}$) and the objective function $f : \mathbb{R}^n \to \mathbb{R}$ are given as follows

$$f(x) \;:=\; \frac{1}{n}\sum_{k=1}^{n} \underbrace{\Bigg(\frac{5}{k} \sum_{\ell=1}^{\frac{n}{2}} \big(\sin x_\ell \cdot \cos x_{n+1-\ell}\big)^2 + 400 \Big(\sum_{\ell=1}^{n} \frac{x_\ell}{\ell} \Big)^2 \Bigg)}_{=:f_k(x)} \tag{5.2}$$

$$\text{for all } x = (x_1,\ldots,x_n) \in \mathbb{R}^n.$$

Since the dimension n is an even natural number, the term $\frac{n}{2}$ is also a natural number. Obviously the functions $f_1,\ldots,f_n$ are non-negative and with a zero 0_n. Thus, the origin 0_n is a global

minimal solution of the optimization problem $\min\limits_{x\in\mathbb{R}^n} f(x)$ with $f(0_n)=0$. With the matrix

$$M := \begin{pmatrix} 1 & \frac{1}{2} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{4} & \cdots & \frac{1}{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{2n} & \cdots & \frac{1}{n^2} \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ \frac{1}{n} \end{pmatrix} \left(1,\ldots,\tfrac{1}{n}\right)$$

we get

$$x^{\mathsf{T}} M x = x^{\mathsf{T}} \begin{pmatrix} 1 \\ \vdots \\ \frac{1}{n} \end{pmatrix} \sum_{\ell=1}^{n} \frac{x_\ell}{\ell} = \left(\sum_{\ell=1}^{n} \frac{x_\ell}{\ell}\right)^2 \text{ for all } x=(x_1,\ldots,x_n)\in\mathbb{R}^n.$$

Hence the last term in the equality (5.2) is a quadratic form with the matrix $400M$ and its gradient is $800Mx$ for all $x\in\mathbb{R}^n$. For every $k\in\{1,\ldots,n\}$ we then get

$$\begin{aligned} \nabla f_k(x) &= \frac{10}{k} \begin{pmatrix} \sin x_1 \cos x_1 \cos^2 x_n \\ \vdots \\ \sin x_{\frac{n}{2}} \cos x_{\frac{n}{2}} \cos^2 x_{\frac{n}{2}+1} \\ -\sin^2 x_{\frac{n}{2}} \cos x_{\frac{n}{2}+1} \sin x_{\frac{n}{2}+1} \\ \vdots \\ -\sin^2 x_1 \cos x_n \sin x_n \end{pmatrix} + 800 \left(\sum_{\ell=1}^{n} \frac{x_\ell}{\ell}\right) \begin{pmatrix} 1 \\ \vdots \\ \frac{1}{n} \end{pmatrix} \\ &\quad \text{for all } x=(x_1,\ldots,x_n)\in\mathbb{R}^n. \end{aligned}$$

Due to the structure of the matrix M, we cannot expect that the Hessian of the functions $f_1,\ldots,f_n$ has equal non-diagonal matrix elements. So it seems to be difficult to choose an appropriate parameter $\alpha\in\mathbb{R}$.

It is obvious that $\bar{x} := (0,\ldots,0)^{\mathsf{T}}$ is a global minimal solution of the optimization problem $\min\limits_{x\in\mathbb{R}^n} f(x)$. Therefore, we choose the starting point $x^0 := \frac{1000}{n}(1,\ldots,1)^{\mathsf{T}}$, which depends on the dimension $n\in\mathbb{N}$. Since the harmonic series $\sum_{\ell=1}^{\infty}\frac{1}{\ell}$ does not converge, its partial sums $\sum_{\ell=1}^{n}\frac{1}{\ell}$ attain large values for high dimensions; hence, the vector $(1,\ldots,1)^{\mathsf{T}}$ must be scaled for use as a starting vector.

As in Example 5.1, we start with the dimension $n := 1{,}000{,}000$, a mini-batch size $b := 100{,}000$, and a maximal number $j_i := 10$ of iterations per mini-batch. The constant step size is now chosen as $\gamma_i := 5\cdot 10^{-4}$ and the parameter α is now set to $\alpha := 10^{-5}$. Numerical results of Algorithm 3.1 obtained with MATLAB R2024b on an 8-core iMac are illustrated in Figure 2. The CPU time required is 9.59 seconds. Table 2 lists results for different dimensions. Note that for 200 million variables, the objective function value at the starting point is already $3.8786\cdot 10^{-6}$, while at the final iteration point the objective function value is $1.2307\cdot 10^{-9}$.

The question now is whether we can get better results for another quasi-Newton parameter $\alpha\neq 10^{-5}$. Varying this parameter in the case of a constant dimension $n := 1{,}000{,}000$, a constant mini-batch size $b := 100{,}000$, a constant step size $\gamma_i := 5\cdot 10^{-4}$, and a constant maximal iteration number $i_{\max} := 10$, we get the results in Table 3. From these results, it is clear that the choice of the quasi-Newton parameter α does not have much impact on the CPU time in this example.

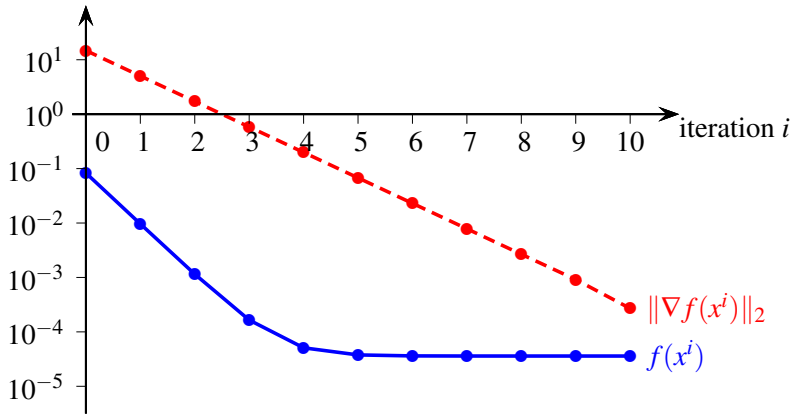

FIGURE 2. Semilogarithmic illustration of the function values $f(x^i)$ and the norms $\|\nabla f(x^i)\|_2$ at the iteration points x^i including the starting point (with $i \in \{0,\ldots,10\}$) in Example 5.2 for the dimension $n = 1{,}000{,}000$.

number n of variables	mini-batch size b	constant step size γ_i	max. iter. no. $i_{\max}$	$f(x_{i_{\max}})$	$\|\nabla f(x_{i_{\max}})\|_2$	CPU time in seconds
1,000,000	100,000	$5 \cdot 10^{-4}$	10	$3.5967 \cdot 10^{-5}$	$2.6951 \cdot 10^{-4}$	9.59
10,000,000	100,000	$5 \cdot 10^{-4}$	10	$4.1737 \cdot 10^{-7}$	$3.6067 \cdot 10^{-5}$	90.27
100,000,000	100,000	$5 \cdot 10^{-4}$	10	$4.7495 \cdot 10^{-9}$	$4.0846 \cdot 10^{-6}$	1,088.52
200,000,000	100,000	$5 \cdot 10^{-4}$	10	$1.2307 \cdot 10^{-9}$	$2.1163 \cdot 10^{-6}$	2,797.76

TABLE 2. Numerical results for Algorithm 3.1 in Example 5.2 using MATLAB R2024b on an 8-core iMac and a quasi-Newton parameter $\alpha = 10^{-5}$.

quasi-Newton parameter α	$f(x_{i_{\max}})$	$\|\nabla f(x_{i_{\max}})\|_2$	CPU time in seconds
10^{-2}	$3.5973 \cdot 10^{-5}$	$3.4673 \cdot 10^{-4}$	9.54
10^{-4}	$3.5974 \cdot 10^{-5}$	$3.2453 \cdot 10^{-4}$	9.59
10^{-6}	$3.5971 \cdot 10^{-5}$	$3.0119 \cdot 10^{-4}$	9.41
10^{-8}	$3.6487 \cdot 10^{-5}$	$1.4844 \cdot 10^{-4}$	9.55
10^{-10}	$3.5939 \cdot 10^{-5}$	$1.137 \cdot 10^{-4}$	9.52
10^{-12}	$3.5934 \cdot 10^{-5}$	$1.1561 \cdot 10^{-4}$	9.39

TABLE 3. Numerical results for Algorithm 3.1 in Example 5.2 for fixed dimension $n := 1{,}000{,}000$ using MATLAB R2024b on an 8-core iMac with fixed parameters $b := 100{,}000$, $\gamma_i := 5 \cdot 10^{-4}$, and $i_{\max} := 10$.

A possible explanation could be that the calculation of the numbers $\lambda_1,\ldots,\lambda_n$ in Remark 3.2 as part of the diagonal elements $\lambda_1 + \alpha,\ldots,\lambda_n + \alpha$ leads to a certain robust "adaptation".

CONCLUSION

The algorithm of this paper combines a mini-batch method and a new simplified quasi-Newton method with the advantage that a solution of highly nonlinear unconstrained optimization problems can be approximated in each mini-batch. As a consequence, the size of the mini-batches used can generally be larger than normal, resulting in reduced gradient noise. Two examples show that the initial optimization problems can be successfully solved even with a very large number of variables. At least in a more complex example, it turns out that the chosen quasi-Newton parameter does not seem to have much impact on CPU times. As is common with stochastic gradient methods, the step sizes chosen for each mini-batch must be carefully set. A numerical comparison of the presented algorithm with current methods requires further investigation.

REFERENCES

[1] M. Zhao, Z. Lai, and L.-H. Lim, Stochastic Steffensen method, Comput. Optim. Appl. 89 (2024), 1–32.
[2] J.F. Steffensen, Remarks on iteration, Skand. Aktuarietidskr. 1 (1933), 64–72.
[3] J.F. Steffensen, Further remarks on iteration, Skand. Aktuarietidskr. 1–2 (1945), 44–55.
[4] J. Barzilai and J.M. Borwein, Two-point step size gradient methods, IMA J. Numer. Anal. 8 (1988), 141–148.
[5] S. Becker, J. Fadili, and P. Ochs, On quasi-Newton forward-backward splitting: proximal calculus and convergence, SIAM J. Optim. 29 (2019), 2445–2481.
[6] J.D. Griffin, M. Jahani, M. Takáč, S. Yektamaram, and W. Zhou, A minibatch stochastic Quasi-Newton method adapted for nonconvex deep learning problems, Optimization Online (2022).
[7] T.-D. Guo, Y. Liu, and C.-Y. Han, An overview of stochastic quasi-Newton methods for large-scale machine learning, J. Oper. Res. Soc. China 11 (2023), 245–275.
[8] L. Bottou, F.E. Curtis, and J. Nocedal, Optimization methods for large-scale machine learning, SIAM Rev. 60 (2018), 223–311.
[9] N.N. Schraudolph, J. Yu, and S. Günter, A stochastic quasi-Newton method for online convex optimization, J. Mach. Learn. Res. 2 (2007), 436–443.
[10] J. Sohl-Dickstein, B. Poole, and S. Ganguli, Fast large-scale optimization by unifying stochastic gradient and quasi-Newton methods, arXiv (2014), accessed October 23, 2024, https://doi.org/10.48550/arXiv.1311.2115 (2014).
[11] X. Wang, S. Ma, D. Goldfarb, and W. Liu, Stochastic quasi-Newton methods for nonconvex stochastic optimization, SIAM J. Optim. 27 (2017), 927–956.
[12] S. Mahboubi, S. Indrapriyadarsini, H. Ninomiya, and H. Asai, Momentum acceleration of quasi-Newton training for neural networks, In: A.C. Nayak and A. Sharma (eds.), PRICAI 2019: Trends in Artificial Intelligence, pp. 268–281, Springer, Cham, 2019.
[13] M. Yang, D. Xu, H. Chen, Z. Wen, and M. Chen, Enhance curvature information by structured stochastic quasi-Newton methods, arXiv (2021), accessed October 23, 2024, https://doi.org/10.48550/arXiv.2006.09606 (2021).
[14] M. Yang, A. Milzarek, Z. Wen, and T. Zhang, A stochastic extra-step quasi-Newton method for nonsmooth nonconvex optimization, Math. Program. 194 (2022), 257–303.
[15] J. Rafati and R.F. Marica, Quasi-Newton optimization methods for deep learning applications, In: M.A. Wani, M. Kantardzic, and M. Sayed-Mouchaweh (eds.), Deep learning applications, pp. 9–38, Springer, Singapore, 2020.
[16] S. Gondaliya and K. Amin, Multi-batch quasi-Newton method with artificial neural network for movie recommendation, J. Inst. Eng. India Ser. B 102 (2021), 729–742.
[17] T. Hong, T.-a. Pham, I. Yavneh, and M. Unser, A mini-batch quasi-Newton proximal method for constrained total-variation nonlinear image reconstruction, arXiv (2024), accessed October 23, 2024, https://doi.org/10.48550/arXiv.2307.02043 (2024).

[18] R. Gower, N. Le Roux, and F. Bach, Tracking the gradients using the Hessian: A new look at variance reducing stochastic methods, In: A. Storkey and F. Perez-Cruz (eds.), Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics, pp. 707–715, PMLR, 2018.
[19] M. Reyad, A.M. Sarhan, and M. Arafa, A modified Adam algorithm for deep neural network optimization, Neural Comput. Appl. 35 (2023), 17095–17112.
[20] S. Li, T. Zhang and Y. Xia, A family of Barzilai-Borwein steplengths from the viewpoint of scaled total least squares, Comput. Optim. Appl. 87 (2024), 1011–1031.
[21] M. Menickelly and S.M. Wild, Stochastic average model methods, Comput. Optim. Appl. 88 (2024), 405–442.
[22] A. Sadiev, A. Beznosikov, A.J. Almansoori, D. Kamzolov, R. Tappenden, and M. Takáč, Stochastic Gradient Methods with Preconditioned Updates, J. Optim. Theory Appl. 201 (2024), 471–489.
[23] S. Becker and M.J. Fadili, A quasi-Newton proximal splitting method, In: F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (eds.), Advances in Neural Information Processing Systems 25 (NIPS 2012), Curran Associates, Inc., 2012.
[24] S. Karimi and S. Vavasis, IMRO: a proximal quasi-Newton method for solving ℓ_1-regularized least squares problems, SIAM J. Optim. 27 (2017), 583–615.
[25] J. Jahn, Order analysis, deep learning, and connections to optimization, Springer, Cham, 2024.
[26] C. Geiger and C. Kanzow, Numerische Verfahren zur Lösung unrestringierter Optimierungsaufgaben, Springer, Berlin, 1999.
[27] J.E. Dennis and J.J. Moré, A characterization of superlinear convergence and its application to quasi-Newton methods, Math. Comp. 28 (1974), 549–560.
[28] W.A. Gruver and E. Sachs, Algorithmic methods in optimal control, Pitman, Boston, 1981.
[29] G. Garrigos and R.M. Gower, Handbook of convergence theorems for (stochastic) gradient methods, arXiv (2023), accessed August 30, 2023, https://arxiv.org/abs/2301.11235v2 (2023).
[30] A. Mokhtari and A. Ribeiro, RES: Regularized stochastic BFGS algorithm, IEEE Trans. Signal Process. 62 (2014), 6089–6104.
[31] X. Zhou, On the Fenchel duality between strong convexity and Lipschitz continuous gradient, arXiv (2018), accessed July 9, 2025, https://arxiv.org/abs/1803.06573.