MATHEMATICS OF COMPUTATION
Volume 00, Number 0, Pages 000–000
S 0025-5718(XX)0000-0

LSOS: LINE-SEARCH SECOND-ORDER STOCHASTIC OPTIMIZATION METHODS FOR NONCONVEX FINITE SUMS

DANIELA DI SERAFINO, NATAŠA KREJIĆ, NATAŠA KRKLEC JERINKIĆ, AND MARCO VIOLA

Abstract. We develop a line-search second-order algorithmic framework for minimizing finite sums. We do not make any convexity assumptions, but require the terms of the sum to be continuously differentiable and have Lipschitz-continuous gradients. The methods fitting into this framework combine line searches and suitably decaying step lengths. A key issue is a two-step sampling at each iteration, which allows us to control the error present in the line-search procedure. Stationarity of limit points is proved in the almost-sure sense, while almost-sure convergence of the sequence of approximations to the solution holds with the additional hypothesis that the functions are strongly convex. Numerical experiments, including comparisons with state-of-the art stochastic optimization methods, show the efficiency of our approach.

1. Introduction

We are interested in solving large finite-sum minimization problems, where the objective function is, e.g., the sample mean of a finite family of possibly nonconvex smooth functions. This is the case of many statistical learning problems, including deep learning and more generally machine learning problems (see, e.g., [16, 7]), which have received much attention in the last years. Specifically, we consider problems of the form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ \phi(\mathbf{x}), \quad \phi(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\phi_i(\mathbf{x}), \tag{1.1}$$

where $\phi(\mathbf{x})$ is bounded from below and each $\phi_i(\mathbf{x})$ is twice continuously differentiable with Lipschitz-continuous gradient. Assuming that N is large, the computation of the objective function and its gradient and Hessian is expensive, and approximations of them are generally used. Subsampling is a natural way of computing these approximations, i.e., for a randomly chosen subsample $\mathcal{M} \subset \{1, \ldots, N\}$ with

2020 *Mathematics Subject Classification.* Primary 65K05, 90C15, 62L20.

Key words and phrases. Nonconvex finite sums, stochastic optimization methods, line search, almost-sure convergence, quasi-Newton methods.

Research supported by the Executive Programme of Scientific and Technological Cooperation between the Italian Republic and the Republic of Serbia for years 2019-21 (Italian grant no. RS19MO05, Serbian grant no. 451-03-9/2021-14/200125). D. di Serafino and M. Viola were also supported by the Istituto Nazionale di Alta Matematica - Gruppo Nazionale per il Calcolo Scientifico (INdAM-GNCS) and by the 2019 V:ALERE Program of the University of Campania "L. Vanvitelli".

cardinality M we can approximate $\phi(\mathbf{x})$ and its gradient and Hessian with the functions

$$\begin{aligned} f_{\mathcal{M}}(\mathbf{x}) &= \frac{1}{M}\sum_{i\in\mathcal{M}}\phi_i(\mathbf{x}), \\ \mathbf{g}_{\mathcal{M}}(\mathbf{x}) &= \frac{1}{M}\sum_{i\in\mathcal{M}}\nabla\phi_i(\mathbf{x}), \\ B_{\mathcal{M}}(\mathbf{x}) &= \frac{1}{M}\sum_{i\in\mathcal{M}}\nabla^2\phi_i(\mathbf{x}), \end{aligned} \tag{1.2}$$

respectively. However, other choices are possible and will be actually used in this work.

The general algorithmic scheme we consider is a second-order scheme with line search. In the deterministic setting, line-search second-order methods are successful in terms of global convergence and convergence rate. Most of these methods preserve their local rate of convergence as the step length tends to be 1 when the iterations come close to the solution. Hence, there is a good tradeoff between benefits and cost, in particular if one applies some second-order method with at least superlinear local convergence. In the stochastic framework the situation is not that simple. A key challenge appears to be the analysis of the gradient error at the new iterate $\mathbf{x}_k + t_k\mathbf{d}_k$, because of the dependence between the step length t_k and the search direction $\mathbf{d}_k$. Since t_k is not fixed a priori, the line search introduces non-martingale dependencies, which do not allow us to easily estimate the error

$$|f_{\mathcal{N}_k}(\mathbf{x}_k + t_k\mathbf{d}_k) - \phi(\mathbf{x}_k + t_k\mathbf{d}_k)|, \tag{1.3}$$

even if $\mathcal{N}_k$ is generated as an independent identically distributed (i.i.d.) sample at each iteration. Thus, we propose a class of methods where this error is controlled by additional sampling. For other possibilities, see, e.g., [20] and [24].

Stochastic optimization methods exploiting search directions based on second-order information have been widely investigated to get better theoretical and practical convergence properties than first-order stochastic methods, for either finite sums or general stochastic problems, especially when badly-scaled problems must be solved. Stochastic versions of Newton-type methods are discussed, e.g., in [2, 6, 9, 10, 12, 33, 34, 35, 36], and variants of the adaptive cubic regularization schemes are proposed in [1, 30]. In particular, stochastic quasi-Newton methods are analyzed, either in the strongly convex or in the nonconvex setting, in [4, 9, 8, 11, 17, 21, 26, 27, 28, 37, 38].

For strongly convex functions of type (1.1), in [28] Moritz et al. propose a stochastic L-BFGS algorithm based on the same inverse Hessian approximation as in [8], but use SVRG [22] instead of the standard stochastic gradient approximation (see, e.g., [7]). This L-BFGS algorithm, which applies a constant step length, has Q-linear rate of convergence of the expected value of the error in the objective function. A modification to this L-BFGS scheme is proposed by Gower et al. in [17], where a stochastic block BFGS update is used, in which the vector pairs for updating the inverse Hessian are replaced by matrix pairs gathering directions and matrix-vector products between subsampled Hessians and those directions. The resulting algorithm uses a constant step length and has Q-linear convergence rate of the expected value of the objective function error, as in the previous case, but appears more efficient by numerical experiments. More recently, the Incremental Quasi-Newton (IQN) method [25] has been developed, which has been proved to have local superlinear convergence. IQN differs from other stochastic quasi-Newton methods as it uses aggregated information on variables, gradients, and quasi-Newton Hessian approximations to reduce the noise of gradient and Hessian approximations, applies a cyclic scheme to update the functions, and approximates each individual function by a Taylor's expansion where the linear and quadratic terms are evaluated with respect to the same iterate. Nevertheless, despite we do not use such a combination of techniques, we will show that the stochastic L-BFGS method presented in this work, which exploits the Jacobian sketching described in [18], in practice is often faster than IQN on widely used test problems.

When dealing with nonconvex finite sums, the (L-)BFGS updates may lead to indefinite Hessian approximations, which in turn may prevent convergence. To cope with this problem, specifically suited versions of stochastic L-BFGS algorithms were developed – see, e.g., [4, 37]. In particular, in [37] the authors propose to use damping to guarantee positive definiteness of the Hessian approximations, and combine the modified L-BFGS update with the SVRG gradient approximation. The resulting algorithm, named SdLBFGS-VR, equipped with a constant step length, is guaranteed to bring the expected value of the gradient norm (computed among all the iterations) below a predefined threshold in a finite number of steps. This method is used for comparison purposes here.

Contribution and outline of this work. We propose a Line-search Second-Order Stochastic (LSOS) algorithmic framework for problem (1.1), where Newton and quasi-Newton directions in a rather broad meaning are used. Inexactness is allowed in the sense that the Newton-type direction can be obtained as inexact solution of the corresponding system of linear equations. The objective function is approximated by subsampling, while other approximations can be used for the gradient and the Hessian. We consider a two-step sampling at each iteration, which allows us to control the error (1.3) introduced by the line-search procedure. The second sampling can be of arbitrary small size, even the sample size 1 is sufficient, and hence it does not increase the computational costs significantly. If the proposed line search is unsuccessful in some prefixed (possibly very large) number of iterations, the algorithm switches to predefined step sizes, i.e., to the SA method.

We prove that limit points are stationary in the almost sure sense for bounded functions with Lipschitz-continuous gradients, while the sequence of approximations to the solution converges almost surely in the case of strongly-convex functions. Although we cannot prove that our class of methods has superlinear convergence rate, we show by numerical experiments that these methods are competitive or faster than state-of-the art second-order stochastic optimization methods. Furthermore, our preliminary experiments show that the line search steps are accepted almost always and the switching point, from the line search to the SA method, is never reached. An additional advantage of the proposed method is that it can be extended to a more general class of problems where the sample is infinite (for example, online training) by performing simple modifications in the LSOS algorithm (see Remark 3.12).

The paper is organized as follows. In Section 2, we introduce the LSOS framework for finite sum problems. In Section 3, we prove almost sure convergence to a stationary point of any algorithm fitting into the general LSOS framework. In Section 4, we present a specialization of the LSOS framework, named LSOS-BFGS, which exploits a mini-batch variant of the SAGA algorithm [13], used in [18], and approximates the inverse Hessian by means of a stochastic version of the limited-memory BFGS (L-BFGS) proposed in [8]. Moreover, for nonconvex objective functions, we propose a modified version of the L-BFGS update in [8], which is inspired by the damping strategy used in [37]. In Section 5, we compare a MATLAB implementation of the LSOS-BFGS algorithm with some state-of-the-art methods in the solution of both nonconvex and convex problems of the form (1.1). Finally, we draw some conclusions in Section 6.

Notation. $\mathbb{E}(x)$ denotes the expectation of a random variable x and $\mathbb{E}(x|\mathcal{F})$ the conditional expectation of x for a given σ-algebra $\mathcal{F}$, where the σ-algebra is determined by random variables precisely defined in the sequel. $\|\cdot\|$ indicates either the Euclidean vector norm or the corresponding induced matrix norm. $\mathbb{R}_+$ and $\mathbb{R}_{++}$ denote the sets of real non-negative and positive numbers, respectively, and $\mathcal{N} = \{1,\dots,N\}$. Vectors are written in boldface and subscripts indicate the elements of a sequence, e.g., $\{\mathbf{x}_k\}$. Furthermore, $\mathbf{g}(\mathbf{x})$ and $B(\mathbf{x})$ denote approximations of the gradient and the Hessian of ϕ at $\mathbf{x}$; we also use $\mathbf{g}_k$ and B_k when $\mathbf{x} = \mathbf{x}_k$. Finally, "a.s." abbreviates "almost sure/surely".

2. Algorithmic framework

We consider second-order methods, where the search direction is found by solving a Newton-type system of linear equations. The dimension of the system may be very large and thus a natural choice is to solve it inexactly, e.g., using some iterative procedure. We consider search directions $\mathbf{d}_k$ such that the following inexact Newton condition is satisfied:

$$\|B_k\mathbf{d}_k + \mathbf{g}_k\| \le \delta_k\|\mathbf{g}_k\|, \tag{2.1}$$

where $\mathbf{g}_k$ may be, e.g., $\mathbf{g}_{\mathcal{N}_k}(x_k)$ for a certain sample $\mathcal{N}_k$. We assume that the approximate gradient $\mathbf{g}_k$ and the approximate Hessian B_k are conditionally independent. This assumption will be discussed later and we will show that quasi-Newton methods satisfy it in the case of finite-sum problems under standard conditions.

Our class of methods fits into the framework of Algorithm 1. It combines the line-search with the Stochastic Approximation (SA) approach [31], where the steplength sequence $\{\alpha_k\}$ is non-summable but square-summable. The main drawback of the latter step lengths is that they quickly become very small and thus the convergence may be extremely slow. On the other hand, the line search introduces non-martingale errors, which are difficult to estimate and bound. The key idea in our algorithm is to combine the two approaches to get a.s. convergence, but keeping the advantage of hopefully large step sizes coming from the line search at least at initial iterations. Additional sampling is performed within the algorithm to control the non-martingale errors.

We comment Algorithm 1 in detail. At the beginning of each iteration $k+1$, the quasi-Newton matrix B_k and the iterate $\mathbf{x}_k$ are available. First, we generate the new sample $\mathcal{N}_k$ and compute $\mathbf{g}_k$. Notice that we do not impose any assumption on $\mathcal{N}_k$ except the unbiasedness and the finite variance of the gradient approximation – see Assumption 3.3 ahead. The search direction $\mathbf{d}_k$ is computed such that (2.2) holds. We then proceed to the step-size computation. The variable *ind* is governing the switch between the line-search step length and the SA step length; as long as $ind = 0$ we perform a line search to get the step size t_k such that the nonmonotone Armijo condition (2.3) is satisfied. That way we get a new candidate point $\overline{\mathbf{x}}_k$. Having

Algorithm 1 Line-search Second-Order Stochastic (LSOS) method

```
 1: given $\mathbf{x}^0 \in \mathbb{R}^n$, $B_0 \in \mathbb{R}^{n\times n}$, $\eta, \beta \in (0,1)$, $\{\alpha_k\}, \{\zeta_k\} \subset \mathbb{R}_{++}$, $\{\delta_k\} \subset \mathbb{R}_+$, $c_{\min}, C_{\max} \in$
    $\mathbb{R}_+$, $K_{\max} \in \mathbb{N}$
 2: $K_f = 0$, $ind = 0$, $k = 0$
 3: while stop criterion not satisfied do
 4:    choose $\mathcal{N}_k \subset \mathcal{N}$ randomly and uniformly and compute $\mathbf{g}_k$
 5:    find a search direction $\mathbf{d}_k$ such that
(2.2)                  $\|B_k\mathbf{d}_k + \mathbf{g}_k\| \leq \delta_k \|\mathbf{g}_k\|$
 6:    if $ind = 0$ then
 7:       find the smallest integer $j \geq 0$ such that the step length $t_k = \beta^j$ satisfies
(2.3)                  $f_{\mathcal{N}_k}(\mathbf{x}_k + t_k\mathbf{d}_k) \leq f_{\mathcal{N}_k}(\mathbf{x}_k) + \eta\, t_k\, \mathbf{g}_k^\top \mathbf{d}_k + \zeta_k$
 8:       $\overline{\mathbf{x}}_k = \mathbf{x}_k + t_k\mathbf{d}_k$
 9:       choose $\mathcal{D}_k \subset \mathcal{N}$ randomly and uniformly
10:       if $f_{\mathcal{D}_k}(\overline{\mathbf{x}}_k) \leq f_{\mathcal{D}_k}(\mathbf{x}_k) - c_{\min}\|\mathbf{g}_{\mathcal{D}_k}(\mathbf{x}_k)\|^2 + C_{\max}\,\zeta_k$ then
11:          $\mathbf{x}_{k+1} = \overline{\mathbf{x}}_k$
12:       else
13:          $\mathbf{x}_{k+1} = \mathbf{x}_k$
14:          $K_f = K_f + 1$
15:          if $K_f > K_{\max}$ then
16:             $ind = 1$
17:          end if
18:       end if
19:    else
20:       $t_k = \alpha_k$
21:       $\mathbf{x}_{k+1} = \mathbf{x}_k + t_k\mathbf{d}_k$
22:    end if
23:    compute $B_{k+1}$.
24:    $k = k + 1$
25: end while
```

the candidate point we perform the additional sampling, generating a sample $\mathcal{D}_k$ uniformly and randomly. Notice that there are no conditions on the size of $\mathcal{D}_k$, i.e., it can be a sample of size 1; then we check if the candidate point $\overline{\mathbf{x}}_k$ satisfies the decrease condition for the approximate function $f_{\mathcal{D}_k}$. In this case, we accept the update $\mathbf{x}_{k+1} = \overline{\mathbf{x}}_k$, the line search is considered to be successful and we proceed to generate the Hessian approximation B_{k+1} that will be used in the next iteration.

In the case of insufficient decrease for the approximate function $f_{\mathcal{D}_k}$, the line search step is not successful, we discard the candidate point $\overline{\mathbf{x}}_k$ and declare $x_{k+1} = x_k$. The counter K_f is increased. As long as $K_f < K_{\max}$ we keep with the line search procedure. However if $K_{\max}$ is reached we change the indicator variable to $ind = 1$ and the algorithm switches to the predefined step sizes α_k, i.e. to the SA method.

The additional check at line 10 is one of the key novelties of the LSOS algorithm and the reasoning behind this check is the following. Under the assumptions stated in the next section, one can prove that the step size t_k in the line search is bounded from below by $t_{\min} := \beta(1-\eta)\mu^2/(L^2(1+\delta_{max})^2)$, provided that $\delta_k \leq \mu/(2L)$ and $\mathbf{g}_k = \mathbf{g}_{\mathcal{N}_k}(\mathbf{x}_k)$. In that case, there holds

$$f_{\mathcal{N}_k}(\overline{\mathbf{x}}_k) \leq f_{\mathcal{N}_k}(\mathbf{x}_k) - c\|\mathbf{g}_k\|^2,$$

where $c := t_{\min}\eta/(2L)$. In other words, this line search provides sufficient decrease with respect to $f_{\mathcal{N}_k}$. Thus, we check if this new point would be of similar quality to the independently chosen subsample $\mathcal{D}_k$ and the corresponding function $f_{\mathcal{D}_k}$ with a deterioration controlled by ζ_k. Implicitly, we can consider this as a check of similarity of the functions f_i. If these functions are similar, then performing a line search on a subsampled function is probably beneficial since it is a good, but cheaper, approximation of the objective function ϕ. Otherwise, the dissimilarity of the functions is too big and the line-search step is not successful. If the unsuccessful steps occur sufficiently many times (K_{max} times), the algorithm is switched to the SA phase, which may be slower, but more stable for that kind of problems. This reasoning helps us to take the advantages of both the SA and LS variants, having a theoretically sound algorithm with good practical behavior.

We note that both decrease rules (lines 7 and 10 of the algorithm) are nonmonotone. The reason is that the inexact direction $\mathbf{d}_k$ does not need to be a decreasing direction for $f_{\mathcal{N}_k}$, but it might be a good direction for the original objective function. The term ζ_k makes the step always well defined as for t_k small enough one can always satisfy the condition (2.3) and hence we have the finite termination of the backtracking loop, see Lemma 2.1 below. Thus the algorithm is well defined. In the case $\mathbf{d}_k$ is a decreasing direction, the nonmonotone rule allows us to take larger steps. As regards the decrease condition at line 10, we notice that the lack of decrease in $f_{\mathcal{D}_k}$ is not necessarily an increase in the function value and given that $C_{\max}, \zeta_k > 0$, we can regard the condition at line 10 of Algorithm 1 as a nonmonotone line-search condition. However, as specified in the next section, we have $\zeta_k \to 0$ and thus this condition becomes stricter and stricter.

Lemma 2.1. *Let f_i, $i = 1, \dots, N$, be continuous. Given $\mathbf{d}_k, \mathbf{g}_k \in \mathbb{R}^n$, and $\zeta_k > 0$ the backtracking defined at line 7 of the LSOS algorithm has finite termination.*

Proof. The function $f_{\mathcal{N}_k}$ is continuous, $\zeta_k > 0$ and for j large enough we can always take $t_k = \beta^j$ such that $t_k \mathbf{d}_k^T \mathbf{g}_k$ and $t_k \mathbf{d}_k$ are sufficiently small and therefore $f_{\mathcal{N}_k}(\mathbf{x}_k + t_k \mathbf{d}_k) - f_{\mathcal{N}_k}(\mathbf{x}_k) - \eta t_k \mathbf{d}_k^T \mathbf{g}_k \le \zeta_k$. □

The integer $K_{\max}$ is arbitrarily large, but fixed, and controls the maximum number of iterations in which we might have an increase in the additional sampled function $f_{\mathcal{D}_k}$, i.e. the maximal number of unsuccessful line-search iterations. At the end of each iteration, we compute the new approximate Hessian B_{k+1}.

In the next section we show that this algorithm converges almost surely. Summarizing the properties of the algorithm, we can see that there are two possible scenarios. The first one is that the number of unsuccessful line-search steps is smaller than $K_{\max}$ when the stopping criteria is reached and hence the method has generated an iterative sequence with the line-search procedure. The second possibility is that $K_f = K_{\max}$ before the stopping criteria is reached and we have that, starting from a given point, the iterate sequence has been generated by SA. Clearly, the role of $K_{\max}$ is essential as its value determines the properties of the sequence generated by LSOS. We observe that it is very likely that a suitable value of $K_{\max}$ can be problem dependent. Nevertheless, by choosing a relatively large $K_{\max}$, one can enforce line-search steps which yield larger step sizes if successful and hence faster convergence. Of course it might happen that many line-search steps are unsuccessful, thus resulting in a waste of time for the algorithm. Anyway, all our experiments, see Section 5, indicate that the line search iterations are successful in a vast majority of cases, the number of discarded candidate points is in the interval 1-6%, and $K_{\max}$ is never reached.

An additional question that may arise here regards the second scenario. Assume that $K_{\max}$ is reached and thus we switch to the SA step sizes. A number of successful methods with approximate gradients and variance reduction are defined for strongly convex problems (see, e.g., [22, 13, 18]). Some of these methods work with fixed step sizes and Algorithm 1 implies decreasing step sizes. Clearly it would be better to use a fixed step size in this scenario, assuming that the search direction $\mathbf{d}_k$ satisfies the variance-reduction properties. One could reformulate Algorithm 1 such that it covers this possibility. However such a reformulation would imply specifying a number of additional assumptions on the gradient approximation as well as on the construction of the Hessian approximation B_k. Our intention here was to propose a rather general scheme, so we did not consider this case separately. But we tested the method against the fixed step-size methods with variance reduction in Section 5 and demonstrated the advantages of the proposed algorithm. Given that we never came close to $K_{\max}$, the second scenario did not occur in our tests.

It is worth mentioning that in [29] and [3] the authors also consider line-search procedures in the stochastic framework. In both cases the key assumptions are that the sequences of random estimates of the function and the gradient are probabilistically accurate in the submartingale sense. In the case of subsampled functions and gradients the condition reduces to taking the size of $\mathcal{N}_k$ large enough to be able to satisfy the required accuracy for both the function and the gradient. Moreover, we note that in both cases, while an Armijo-like condition is checked (in [3] the authors use a nonmonotone condition relying on an a-priori knowledge of the function evalution accuracy), no formal backtracking is performed. In fact, if the line-search condition is not satisfied by the initial step length, then the authors propose to immediately reject the iterate, shrink the step length and recompute the gradient and function approximations from scratch. These ingredients allow the authors to develop stochastic Armijo-like line-search methods that need neither the additional sample $\mathcal{D}_k$ nor switching to SA in any scenario. We note that the sample size fulfilling the aforementioned probabilistic assumptions is rather large while we do not impose any condition on $\mathcal{N}_k$ besides unbiasedeness and finite variance of the gradient approximation. Hence, it is quite difficult to compare the approach we propose here with the ones in [29, 3]. Furthermore, the complexity results given in the mentioned works rely on fixed probabilities of the estimated function and gradient, while we do not present formal complexity results here. To offer some insight into the complexity of the proposed method, in particular with respect to the possible increase of the oracle complexity due to the independent sampling of $\mathcal{N}_k$ and $\mathcal{D}_k$, in Section 5 we provide an empirical analysis in terms of oracle complexity for LSOS, comparing it with state-of-the-art methods in terms of number of data accesses.

3. Convergence theory

We state more formally our assumptions on the minimization problem and on some quantities used in Algorithm LSOS.

Assumption 3.1. The function ϕ is bounded from below by ϕ^* and the functions ϕ_i have Lipschitz-continuous gradients with Lipschitz constant L.

Although we do not suppose ϕ is strongly convex, we make the following assumption on the approximate Hessians computed by the algorithm. Without loss of generality we take the same L as in the previous assumption.

Assumption 3.2. There exist $\mu, L > 0$ such that

$$\mu I \preceq B_k \preceq LI$$

for all k.

We also specify some properties of the gradient approximation. To this aim, we denote by $\boldsymbol{\varepsilon}_g(\mathbf{x})$ the error in the approximation of $\nabla\phi(\mathbf{x})$:

$$\nabla\phi(\mathbf{x}) = \mathbf{g}(\mathbf{x}) + \boldsymbol{\varepsilon}_g(\mathbf{x}), \tag{3.1}$$

and by $\mathcal{F}_k$ the σ-algebra identified by $\{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_k\}$.

Assumption 3.3. There exists a constant $\bar{c}_1 > 0$ such that

$$\mathbb{E}(\mathbf{g}_k(\mathbf{x}_k)|\mathcal{F}_k) = \nabla\phi(\mathbf{x}_k) \text{ and } \mathbb{E}(\|\varepsilon_{g_k}(\mathbf{x}_k)\|^2|\mathcal{F}_k) \le \bar{c}_1. \tag{3.2}$$

for all k.

In other words, we assume that the expected gradient noise is zero and the variance of the gradient errors is bounded.

Finally, we make some (standard) assumptions on the sequences $\{\alpha_k\}, \{\zeta_k\} \subset \mathbb{R}_{++}$ and $\{\delta_k\} \subset \mathbb{R}_+$.

Assumption 3.4.

$$\begin{array}{ll} \displaystyle\sum_{k=1}^{\infty} \zeta_k < \infty, & \\ \displaystyle\sum_{k=0}^{\infty} \alpha_k = \infty, & \displaystyle\sum_{k=1}^{\infty} \alpha_k^2 < \infty, \\ \delta_k \to 0, & \displaystyle\sum_{k=1}^{\infty} \delta_k \alpha_k < \infty. \end{array}$$

Let us recall a few properties of the algorithm here. In the first iterations of the algorithm, non-descent directions are likely to occur; however, by requiring $\zeta_k > 0$ we ensure that the line search remains well defined, by Lemma 2.1. Furthermore, by Assumption 3.4 it is $\zeta_k \to 0$, which implies that Algorithm 1 eventually determines a descent direction for the current approximation of the objective function. At the k-th iteration, we may reject the update $\overline{\mathbf{x}}_{k+1} = \mathbf{x}_k + t_k\mathbf{d}_k$ obtained with the line search, if the condition at line 10 of the algorithm is not satisfied. However, only a finite number of rejections, specified by $K_{\max}$, is allowed. In any case the new approximate Hessian B_{k+1} is computed at the end. The $(k+1)$-st iteration starts with sampling at line 4, and the sample $\mathcal{N}_{k+1}$ is used to compute the approximations of the gradient and the function. Thus B_{k+1} and $\mathbf{g}_{k+1}$ are conditionally independent in the sense specified by the following lemma.

Lemma 3.5. *The approximate gradient $\mathbf{g}_k$ and the approximate Hessian B_k generated by Algorithm 1 satisfy*

$$\mathbb{E}(B_k\mathbf{g}_k|\mathcal{F}_k) = B_k\mathbb{E}(\mathbf{g}_k|\mathcal{F}_k).$$

Next, we give some properties of the directions $\mathbf{d}_k$ generated by Algorithm 1. Throughout this section we use $\delta_{\max} = \max_k \delta_k$.

Lemma 3.6. *Suppose that Assumption 3.2 holds and $\delta_k \leq \mu/(2L)$. Then*

$$\mathbf{g}_k^\top \mathbf{d}_k \leq -\frac{1}{2L}\|\mathbf{g}_k\|^2 \tag{3.3}$$

and

$$\|\mathbf{d}_k\|^2 \leq \frac{(\delta_{max}+1)^2}{\mu^2}\|\mathbf{g}_k\|^2. \tag{3.4}$$

Proof. Let $\mathbf{r}_k = B_k\mathbf{d}_k + \mathbf{g}_k$. We have

$$\mathbf{d}_k = B_k^{-1}\mathbf{r}_k - B_k^{-1}\mathbf{g}_k. \tag{3.5}$$

Assumption 3.2 together with (2.2) implies

$$\begin{aligned} \mathbf{g}_k^\top \mathbf{d}_k &= \mathbf{g}_k^\top B_k^{-1}\mathbf{r}_k - \mathbf{g}_k^\top B_k^{-1}\mathbf{g}_k \\ &\leq \|\mathbf{g}_k\|\|B_k^{-1}\|\|\mathbf{r}_k\| - \frac{1}{L}\|\mathbf{g}_k\|^2 \\ &\leq \frac{1}{\mu}\delta_k\|\mathbf{g}_k\|^2 - \frac{1}{L}\|\mathbf{g}_k\|^2 \\ &= \left(\frac{\delta_k}{\mu} - \frac{1}{L}\right)\|\mathbf{g}_k\|^2. \end{aligned}$$

Since $\delta_k \leq \frac{\mu}{2L}$, we conclude that (3.3) holds. Furthermore, for each k we have

$$\|\mathbf{d}_k\| = \|B_k^{-1}(\mathbf{r}_k - \mathbf{g}_k)\| \leq \frac{1}{\mu}(\|\mathbf{r}_k\| + \|\mathbf{g}_k\|) \leq \frac{\delta_k+1}{\mu}\|\mathbf{g}_k\|, \tag{3.6}$$

thus, squaring and using $\delta_k \leq \delta_{\max}$ we obtain (3.4). □

The following theorem on the convergence of perturbed nonnegative supermartingales [32] is used for proving the convergence when the step-length choice corresponding to *ind* $= 1$ is activated.

Theorem 3.7. *Let $U_k, \beta_k, \xi_k, \rho_k \geq 0$ be $\mathcal{F}_k$-measurable random variables such that*

$$\mathbb{E}(U_{k+1}|\mathcal{F}_k) \leq (1+\beta_k)U_k + \xi_k - \rho_k, \quad k = 1, 2, \ldots .$$

If $\sum_k \beta_k < \infty$ and $\sum_k \xi_k < \infty$, then $U_k \to U < \infty$ a.s. and $\sum_k \rho_k < \infty$ a.s..

Now we are ready to prove convergence results for Algorithm LSOS. Given that two scenarios are possible, i.e. $K_{\max}$ is reached or not, we consider the following two theorems and then state the overall convergence result by combining them.

Theorem 3.8. *Let Assumptions 3.1-3.4 hold, $\{\mathbf{x}_k\}$ be a sequence generated by Algorithm 1 and assume that $K_f = K_{\max}+1$ is reached. Then*

$$\liminf_{k\to\infty} \|\nabla\phi(\mathbf{x}_k)\| = 0 \quad a.s.. \tag{3.7}$$

Proof. Given that $K_f > K_{max}$, the SA step length is eventually chosen. Then there exists $\overline{k}$ such that the SA step length is chosen for all $k \geq \overline{k}$. Assumption 3.3 implies

$$\begin{aligned} \mathbb{E}(\|\mathbf{g}_k\|^2|\mathcal{F}_k) &\leq 2\left(\mathbb{E}(\|\nabla\phi(\mathbf{x}_k)\|^2|\mathcal{F}_k) + \mathbb{E}(\|\varepsilon_{g_k}(\mathbf{x}_k)\|^2|\mathcal{F}_k)\right) \\ &\leq 2\left(\|\nabla\phi(\mathbf{x}_k)\|^2 + \overline{c}_1\right). \end{aligned} \tag{3.8}$$

Moreover, we have

$$\begin{aligned} \mathbb{E}(\|\mathbf{g}_k\||\mathcal{F}_k) &\leq \sqrt{\mathbb{E}(\|\mathbf{g}_k\|^2|\mathcal{F}_k)} \leq \sqrt{2\left(\|\nabla\phi(\mathbf{x}_k)\|^2 + \overline{c}_1\right)} \\ &\leq \sqrt{2}\left(\|\nabla\phi(\mathbf{x}_k)\| + \sqrt{\overline{c}_1}\right). \end{aligned}$$

Recall that for all $k \geq \overline{k}$ we have $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{d}_k$, where α_k is pre-determined and $\mathbf{d}_k$ satisfies (2.2) in algorithm 1. From now on, we assume $k \geq \overline{k}$. Notice that Assumptions 3.2 and 3.3 together with Lemma 3.5 applied to B_k^{-1} imply

$$\begin{aligned} \nabla\phi(\mathbf{x}_k)^\top\mathbb{E}(B_k^{-1}\mathbf{g}_k|\mathcal{F}_k) &= \nabla\phi(\mathbf{x}_k)^\top B_k^{-1}\mathbb{E}(\mathbf{g}_k|\mathcal{F}_k) \\ &= \nabla\phi(\mathbf{x}_k)^\top B_k^{-1}\nabla\phi(\mathbf{x}_k) \geq \frac{1}{L}\|\nabla\phi(\mathbf{x}_k)\|^2. \end{aligned} \tag{3.9}$$

This and (3.5) lead to

$$\begin{aligned} \mathbb{E}(\alpha_k\nabla\phi(\mathbf{x}_k)^\top\mathbf{d}_k|\mathcal{F}_k) &= \alpha_k\nabla\phi(\mathbf{x}_k)^\top\mathbb{E}(\mathbf{d}_k|\mathcal{F}_k) \\ &= \alpha_k\nabla\phi(\mathbf{x}_k)^\top\mathbb{E}(B_k^{-1}\mathbf{r}_k - B_k^{-1}\mathbf{g}_k|\mathcal{F}_k) \\ &\leq \alpha_k\left(\|\nabla\phi(\mathbf{x}_k)\|\mathbb{E}(\|B_k^{-1}\|\|\mathbf{r}_k\||\mathcal{F}_k) - \frac{1}{L}\|\nabla\phi(\mathbf{x}_k)\|^2\right) \\ &\leq \alpha_k\left(\|\nabla\phi(\mathbf{x}_k)\|\mathbb{E}(\frac{\delta_k}{\mu}\|\mathbf{g}_k\||\mathcal{F}_k) - \frac{1}{L}\|\nabla\phi(\mathbf{x}_k)\|^2\right) \\ &\leq \alpha_k\|\nabla\phi(\mathbf{x}_k)\|^2\left(\frac{\sqrt{2}\delta_k}{\mu} - \frac{1}{L}\right) + \alpha_k\|\nabla\phi(\mathbf{x}_k)\|\frac{\sqrt{2\,\overline{c}_1}\delta_k}{\mu}. \end{aligned} \tag{3.10}$$

Using Assumption 3.1 and the descent lemma [5, Proposition A24], we get

$$\phi(\mathbf{x}_{k+1}) \leq \phi(\mathbf{x}_k) + \alpha_k\nabla\phi(\mathbf{x}_k)^\top\mathbf{d}_k + \frac{1}{2}\alpha_k^2 L\,\|\mathbf{d}_k\|^2.$$

Applying the conditional expectation and using (3.10) and (3.4) we obtain

$$\begin{aligned}(3.11)\quad \mathbb{E}(\phi(\mathbf{x}_{k+1})|\mathcal{F}_k) \quad &\leq \quad \mathbb{E}(\phi(\mathbf{x}_k)|\mathcal{F}_k) + \alpha_k \|\nabla\phi(\mathbf{x}_k)\|^2 \left(\frac{\sqrt{2}\delta_k}{\mu} - \frac{1}{L}\right) \\ &\quad +\alpha_k\|\nabla\phi(\mathbf{x}_k)\|\frac{\sqrt{2\,\overline{c}_1}\delta_k}{\mu} \\ &\quad +\frac{1}{2}\alpha_k^2 L\mathbb{E}\left(\frac{(\delta_{max}+1)^2}{\mu^2}\|\mathbf{g}_k\|^2|\mathcal{F}_k\right) \\ &\leq \quad \phi(\mathbf{x}_k) + \alpha_k\|\nabla\phi(\mathbf{x}_k)\|^2\left(\frac{\sqrt{2}\delta_k}{\mu} - \frac{1}{L}\right) \\ &\quad +\alpha_k\|\nabla\phi(\mathbf{x}_k)\|\frac{\sqrt{2\,\overline{c}_1}\delta_k}{\mu} \\ &\quad +\frac{1}{2}\alpha_k^2 L\frac{(\delta_{max}+1)^2}{\mu^2}2(\|\nabla\phi(\mathbf{x}_k)\|^2 + \overline{c}_1).\end{aligned}$$

Rearranging this inequality we obtain

$$(3.12)\ \mathbb{E}(\phi(\mathbf{x}_{k+1})|\mathcal{F}_k) \leq \phi(\mathbf{x}_k) + \|\nabla\phi(\mathbf{x}_k)\|^2\alpha_k(c_k - \frac{1}{L}) + \alpha_k^2\overline{c}_2 + \|\nabla\phi(\mathbf{x}_k)\|\alpha_k\delta_k\overline{c}_3,$$

where

$$c_k = \frac{\sqrt{2}\delta_k}{\mu} + \frac{\alpha_k L(\delta_{max}+1)^2}{\mu^2},\ \overline{c}_2 = \frac{\overline{c}_1 L(\delta_{max}+1)^2}{\mu^2},\ \overline{c}_3 = \frac{\sqrt{2\,\overline{c}_1}}{\mu}.$$

By Assumption 3.4, we have that $c_k \to 0$ and thus there exists $\widetilde{k} \geq \overline{k}$ such that for all $k \geq \widetilde{k}$ we have $c_k - 1/L \leq -1/(2L)$. Henceforth, we assume $k \geq \widetilde{k}$. From (3.12) we obtain

$$(3.13)\quad \mathbb{E}(\phi(\mathbf{x}_{k+1})|\mathcal{F}_k) \leq \phi(\mathbf{x}_k) - \frac{1}{2L}\|\nabla\phi(\mathbf{x}_k)\|^2\alpha_k + \alpha_k^2\overline{c}_2 + \|\nabla\phi(\mathbf{x}_k)\|\alpha_k\delta_k\overline{c}_3,$$

Let us consider two cases: i) $\|\nabla\phi(\mathbf{x}_k)\| \leq 1$, and ii) $\|\nabla\phi(\mathbf{x}_k)\| > 1$. If $\|\nabla\phi(\mathbf{x}_k)\| \leq 1$, then

$$\|\nabla\phi(\mathbf{x}_k)\|\alpha_k\delta_k\overline{c}_3 \leq \alpha_k\delta_k\overline{c}_3.$$

If $\|\nabla\phi(\mathbf{x}_k)\| > 1$, then $\|\nabla\phi(\mathbf{x}_k)\| \leq \|\nabla\phi(\mathbf{x}_k)\|^2$ and

$$\|\nabla\phi(\mathbf{x}_k)\|\alpha_k\delta_k\overline{c}_3 \leq \|\nabla\phi(\mathbf{x}_k)\|^2\alpha_k\delta_k\overline{c}_3.$$

Thus, we can conclude that the following inequality holds in general:

$$\|\nabla\phi(\mathbf{x}_k)\|\alpha_k\delta_k\overline{c}_3 \leq \alpha_k\delta_k\overline{c}_3 + \|\nabla\phi(\mathbf{x}_k)\|^2\alpha_k\delta_k\overline{c}_3,$$

and putting it in (3.13) we get

$$(3.14)\quad \mathbb{E}(\phi(\mathbf{x}_{k+1})|\mathcal{F}_k) \leq \phi(\mathbf{x}_k) - \alpha_k\|\nabla\phi(\mathbf{x}_k)\|^2\left(\frac{1}{2L} - v_k\right) + \alpha_k^2\overline{c}_2 + \alpha_k\delta_k\overline{c}_3,$$

where $v_k = \alpha_k\delta_k\overline{c}_3$. Using Assumption 3.4 we have that $v_k \to 0$ and thus there exists $\hat{k} \geq \widetilde{k}$ such that for all $k \geq \hat{k}$ we have $v_k - 1/(2L) \leq -1/(4L)$. Thus, for all $k \geq \hat{k}$ we have

$$(3.15)\quad \mathbb{E}(\phi(\mathbf{x}_{k+1}) - \phi^*|\mathcal{F}_k) \leq \phi(\mathbf{x}_k) - \phi^* - \alpha_k\frac{1}{4L}\|\nabla\phi(\mathbf{x}_k)\|^2 + \alpha_k^2\overline{c}_2 + \alpha_k\delta_k\overline{c}_3,$$

where ϕ^* is the lower bound of ϕ in Assumption 3.1. Applying Theorem 3.7 with $U_k = \phi(\mathbf{x}_k) - \phi^*, \beta_k = 0, \xi_k = \alpha_k^2 \bar{c}_2 + \alpha_k \delta_k \bar{c}_3$ and $\rho_k = \alpha_k \frac{1}{4L} \|\nabla\phi(\mathbf{x}_k)\|^2$, we conclude that $\phi(\mathbf{x}_k)$ converges a.s. to a value in $[\phi^*, +\infty)$ and

$$\sum_{k=\hat{k}}^{\infty} \alpha_k \|\nabla\phi(\mathbf{x}_k)\|^2 < \infty \quad \text{a.s.}$$

Since $\sum_{k=0}^{\infty} \alpha_k = \infty$, the statement (3.7) holds. □

Theorem 3.9. *Let Assumptions 3.1-3.4 hold, $\{\mathbf{x}_k\}$ be a sequence generated by Algorithm 1 and $K_f \leq K_{max}$. Then*

$$\lim_{k\to\infty} \|\nabla\phi(\mathbf{x}_k)\| = 0 \quad a.s. \tag{3.16}$$

and each limit point of $\{\mathbf{x}_k\}$ is stationary a.s. for problem (1.1).

Proof. Given that $K_f \leq K_{max}$, the SA step length is never used. Then there exists $\bar{k}$ such that for all $k \geq \bar{k}$ we have

$$f_{\mathcal{D}_k}(\mathbf{x}_{k+1}) \leq f_{\mathcal{D}_k}(\mathbf{x}_k) - c_{\min}\|\mathbf{g}_{\mathcal{D}_k}(\mathbf{x}_k)\|^2 + C_{\max}\,\zeta_k.$$

Let us denote $\mathcal{F}_{k+1/2}$ the σ-algebra identified by $\mathcal{N}_0, \mathcal{D}_0, \ldots, \mathcal{N}_{k-1}, \mathcal{D}_{k-1}, \mathcal{N}_k$. Then

$$\mathbb{E}(f_{\mathcal{D}_k}(\mathbf{x}_{k+1})|\mathcal{F}_{k+1/2}) \leq \mathbb{E}(f_{\mathcal{D}_k}(\mathbf{x}_k)|\mathcal{F}_{k+1/2}) - c_{\min}\mathbb{E}(\|\mathbf{g}_{\mathcal{D}_k}(\mathbf{x}_k)\|^2|\mathcal{F}_{k+1/2}) + C_{\max}\,\zeta_k.$$

Furthermore,

$$\mathbb{E}(f_{\mathcal{D}_k}(\mathbf{x}_{k+1})|\mathcal{F}_{k+1/2}) = \phi(\mathbf{x}_{k+1}), \quad \mathbb{E}(f_{\mathcal{D}_k}(\mathbf{x}_k)|\mathcal{F}_{k+1/2}) = \phi(\mathbf{x}_k), \tag{3.17}$$

and

$$\mathbb{E}(\mathbf{g}_{\mathcal{D}_k}(\mathbf{x}_k)|\mathcal{F}_{k+1/2}) = \nabla\phi(\mathbf{x}_k). \tag{3.18}$$

The latter equality implies

$$\begin{aligned}\|\nabla\phi(\mathbf{x}_k)\|^2 = \|\mathbb{E}(\mathbf{g}_{\mathcal{D}_k}(\mathbf{x}_k)|\mathcal{F}_{k+1/2})\|^2 &\leq \mathbb{E}^2(\|\mathbf{g}_{\mathcal{D}_k}(\mathbf{x}_k)\||\mathcal{F}_{k+1/2}) \\ &\leq \mathbb{E}(\|\mathbf{g}_{\mathcal{D}_k}(\mathbf{x}_k)\|^2|\mathcal{F}_{k+1/2}),\end{aligned}$$

and hence

$$-c_{\min}\mathbb{E}(\|\mathbf{g}_{\mathcal{D}_k}(\mathbf{x}_k)\|^2|\mathcal{F}_{k+1/2}) \leq -c_{\min}\|\nabla\phi(\mathbf{x}_k)\|^2.$$

We can conclude that

$$\phi(\mathbf{x}_{k+1}) \leq \phi(\mathbf{x}_k) - c_{\min}\|\nabla\phi(\mathbf{x}_k)\|^2 + C_{\max}\,\zeta_k,$$

and thus, for any $p \in \mathbb{N}$,

$$\phi(\mathbf{x}_{k+p}) \leq \phi(\mathbf{x}_k) - c_{min}\sum_{j=0}^{p-1}\|\nabla\phi(\mathbf{x}_{k+j})\|^2 + C_{max}\sum_{j=0}^{p-1}\zeta_{k+j}.$$

Notice that in this case, when $K_f \leq K_{max}$ for all $k = 0, 1, ...$, we have either

$$f_{\mathcal{D}_k}(\mathbf{x}_{k+1}) \leq f_{\mathcal{D}_k}(\mathbf{x}_k) - c_{\min}\|\mathbf{g}_{\mathcal{D}_k}(\mathbf{x}_k)\|^2 + C_{\max}\,\zeta_k \leq f_{\mathcal{D}_k}(\mathbf{x}_k) + C_{\max}\,\zeta_k$$

or $\mathbf{x}_{k+1} = \mathbf{x}_k$, and thus for all $k = 0, 1, ...$ there holds

$$f_{\mathcal{D}_k}(\mathbf{x}_{k+1}) = f_{\mathcal{D}_k}(\mathbf{x}_k) \leq f_{\mathcal{D}_k}(\mathbf{x}_k) + C_{\max}\,\zeta_k.$$

Applying the conditional expectation $\mathbb{E}(\cdot|\mathcal{F}_{k+1/2})$ we obtain that the following inequality holds for all $k = 0, 1, ...$:

$$\phi(\mathbf{x}_{k+1}) \leq \phi(\mathbf{x}_k) + C_{\max}\,\zeta_k$$

and thus the summability of ζ_k implies that for all $k = 0, 1, \ldots$ we have

$$\phi(\mathbf{x}_{k+1}) \le \phi(\mathbf{x}_0) + C_{\max} \sum_{j=0}^{k} \zeta_j \le \phi(\mathbf{x}_0) + C_{\max} \sum_{j=0}^{\infty} \zeta_j := \overline{c}_4 < \infty$$

i.e., $\phi(\mathbf{x}_k)$ is uniformly bounded from above with a constant $\overline{c}_4$ that depends on C_{max}, ζ_k and $\mathbf{x}_0$. Therefore, we conclude that for any $p \in \mathbb{N}$,

$$\phi(\mathbf{x}_{\overline{k}+p}) \le \overline{c}_4 - c_{min} \sum_{j=0}^{p-1} \|\nabla \phi(\mathbf{x}_{\overline{k}+j})\|^2 + C_{max} \sum_{j=0}^{p-1} \zeta_{\overline{k}+j}.$$

Since ϕ is bounded from below, by taking the expectation, letting p tend to ∞ and using the summability of ζ_k, we get

$$\sum_{k=0}^{\infty} \mathbb{E}(\|\nabla \phi(\mathbf{x}_k)\|^2) < \infty.$$

Finally, by Markov's inequality we have that for any $\epsilon > 0$

$$P(\|\nabla \phi(\mathbf{x}_k)\| \ge \epsilon) \le \frac{\mathbb{E}(\|\nabla \phi(\mathbf{x}_k)\|^2)}{\epsilon^2}$$

and therefore

$$\sum_{k=0}^{\infty} P(\|\nabla \phi(\mathbf{x}_k)\| \ge \epsilon) < \infty.$$

Finally, Borel-Cantelli Lemma [23, Theorem 2.7] implies that $\lim_{k\to\infty} \nabla \phi(\mathbf{x}_k) = 0$ a.s., and by the continuity of $\nabla \phi$ we conclude that every limit point of $\{\mathbf{x}_k\}$ is stationary for ϕ a.s.. □

The overall convergence statement is a simple combination of the previous two theorems as stated below.

Theorem 3.10. *Let Assumptions 3.1-3.4 hold, $\{\mathbf{x}_k\}$ be a sequence generated by Algorithm 1. Then*

$$\liminf_{k\to\infty} \|\nabla \phi(\mathbf{x}_k)\| = 0 \quad \textit{a.s.}.$$

Furthermore, if $K_f \le K_{max}$ then

$$\lim_{k\to\infty} \|\nabla \phi(\mathbf{x}_k)\| = 0 \quad \textit{a.s.}$$

and each limit point of $\{\mathbf{x}_k\}$ is stationary a.s. for problem (1.1).

If $\phi(x)$ is μ-strongly convex we have a stronger convergence result. In this case problem (1.1) has a unique solution $\mathbf{x}^*$.

Theorem 3.11. *Let Assumptions 3.1-3.4 hold and $\{\mathbf{x}_k\}$ be a sequence generated by Algorithm 1. If ϕ is μ-strongly convex, then the sequence $\{\mathbf{x}_k\}$ converges a.s. to the unique solution $\mathbf{x}^*$ of problem* (1.1).

Proof. First, notice that Assumption 3.1 and strong convexity imply

$$\frac{\mu}{2}\|\mathbf{x}_k - \mathbf{x}_*\|^2 \le \phi(\mathbf{x}_k) - \phi(\mathbf{x}_*) \le \frac{L}{2}\|\nabla \phi(\mathbf{x}_k)\|^2, \tag{3.19}$$

where L is as in Assumption 3.1. If $K_f \le K_{max}$, then Theorem 3.9 implies

$$\lim_{k\to\infty} \mathbf{x}_k = \mathbf{x}_* \quad \text{a.s.}.$$

On the other hand, if the SA step length is eventually chosen, from Theorem 3.8 we know that there exists a subsequence $\{\mathbf{x}_k\}_{k \in K \subseteq \mathbb{N}}$ such that

$$\lim_{k \in K} \|\nabla \phi(\mathbf{x}_k)\| = 0 \quad \text{a.s.}.$$

This, together with (3.19), implies

$$\lim_{k \in K} \mathbf{x}_k = \mathbf{x}_* \quad \text{a.s.}.$$

By the continuity of ϕ we get

$$\lim_{k \in K} \phi(\mathbf{x}_k) = \phi(\mathbf{x}_*) \quad \text{a.s.},$$

and according to Theorem 3.7 the whole sequence $\{\phi(\mathbf{x}_k)\}$ converges a.s.. Thus

$$\lim_{k \to \infty} \phi(\mathbf{x}_k) = \lim_{k \in K} \phi(\mathbf{x}_k) = \phi(\mathbf{x}_*) \;\; \text{a.s.},$$

which, together with (3.19), implies the thesis. □

Remark 3.12. It is worth noting that the results proved in Theorems 3.10 and 3.11 can be extended to the more general case given by

$$\phi(\mathbf{x}) = \mathbb{E}(\psi(\mathbf{x}; \boldsymbol{\xi})), \tag{3.20}$$

where $\boldsymbol{\xi} \in \mathbb{R}^m$ is a random vector defined on a probability space. This formulation is usually considered when dealing with infinite samples. In this case, one can approximate the objective function and its derivatives with sample means of the form (1.2), in which $\phi_i(\mathbf{x}) = \psi(\mathbf{x}; \boldsymbol{\xi}_i)$, where $\boldsymbol{\xi}_i$ is a realization of the random vector $\boldsymbol{\xi}$. In this setting, the same convergence results hold provided that (3.17)-(3.18) hold and all the functions ϕ_i are bounded from below.

4. An L-BFGS version of LSOS

Subsampling is a natural way of generating approximations of the objective function and its derivatives in the case of finite-sum problems. According to Algorithm 1, we have that $f_{\mathcal{N}_k}(\mathbf{x}_k)$, $f_{\mathcal{D}_k}(\mathbf{x}_{k+1})$ and $\mathbf{g}_{\mathcal{D}_k}(\mathbf{x}_{k+1})$ are unbiased estimators of $\phi(\mathbf{x}_k)$, $\phi(\mathbf{x}_{k+1})$ and $\nabla \phi(\mathbf{x}_{k+1})$, respectively. The derivative estimate corresponding to the sample $\mathcal{N}_k$ can be replaced by a more sophisticated one, with the aim, e.g., of improving the performance of the method. The Hessian approximation only needs to have eigenvalues that are uniformly bounded from above and away from zero in order to prove the results presented in the previous section. Therefore, our convergence theory still holds if one replaces the subsampled Hessian approximation with suitable quasi-Newton approximations.

In the case of strongly convex problems, Byrd et al. [8] propose to use subsampled gradients and an approximation of the inverse of the Hessian $\nabla^2 \phi(\mathbf{x})$, say H_k, built by means of a stochastic variant of the limited-memory BFGS (L-BFGS) method. Given a memory parameter m, H_k is defined by applying m BFGS updates to an initial matrix, using the m most recent correction pairs $(\mathbf{s}_j, \mathbf{y}_j) \in \mathbb{R}^n \times \mathbb{R}^n$, like in the deterministic version of the L-BFGS method. The pairs are obtained by averaging iterates, i.e., every l steps the following vectors are computed

$$\mathbf{w}_j = \frac{1}{l} \sum_{i=k-l+1}^{k} \mathbf{x}_i, \quad \mathbf{w}_{j-1} = \frac{1}{l} \sum_{i=k-2l+1}^{k-l} \mathbf{x}_i, \tag{4.1}$$

where $j = k/l$, and they are used to build $\mathbf{s}_j$ and $\mathbf{y}_j$ as specified next:

$$\mathbf{s}_j = \mathbf{w}_j - \mathbf{w}_{j-1}, \quad \mathbf{y}_j = B_{\mathcal{T}_j}(\mathbf{w}_j)\,\mathbf{s}_j, \tag{4.2}$$

where $\mathcal{T}_j \subset \mathcal{N}$. By defining the set of the m most recent correction pairs as

$$\{(\mathbf{s}_j, \mathbf{y}_j),\ j = 1, \ldots, m\},$$

the inverse Hessian approximation is computed as

$$H_k = H_k^{(m)}, \tag{4.3}$$

where for $j = 1, \ldots, m$

$$H_k^{(j)} = \left(I - \frac{\mathbf{s}_j\,\mathbf{y}_j^\top}{\mathbf{s}_j^\top \mathbf{y}_j}\right)^\top H_k^{(j-1)} \left(I - \frac{\mathbf{y}_j\,\mathbf{s}_j^\top}{\mathbf{s}_j^\top \mathbf{y}_j}\right) + \frac{\mathbf{s}_j\,\mathbf{s}_j^\top}{\mathbf{s}_j^\top \mathbf{y}_j}, \tag{4.4}$$

and $H_k^{(0)} = (\mathbf{s}_m^\top \mathbf{y}_m / \|\mathbf{y}_m\|^2)\, I$. It can be proved (see [8, Lemma 3.1] and [28, Lemma 4]) that for approximate inverse Hessians of the form (4.3) there exist constants λ_1 and λ_2, with $\lambda_2 \geq \lambda_1 > 0$, such that

$$\lambda_1 I \preceq H_k \preceq \lambda_2 I, \tag{4.5}$$

and hence Assumption 3.2 holds with $\mu = 1/\lambda_2$ and $L = 1/\lambda_1$. The authors of [8] propose a stochastic second-order method for convex problems in which the direction is computed as

$$\mathbf{d}_k = -H_k\,\mathbf{g}_{\mathcal{N}_k}(\mathbf{x}_k),$$

and prove R-linear decrease of the expected value of the error in the function value. For the subsampled gradient and the approximation of the Hessian described above we can easily prove that Lemma 3.5 holds.

In the nonconvex case there is no guarantee that the vectors $\mathbf{s}_j$ and $\mathbf{y}_j$ defined in (4.2) satisfy the condition $\mathbf{s}_j^\top \mathbf{y}_j > 0$, which is needed to preserve the positive definiteness of the inverse Hessian approximation H_k. Therefore, we propose to modify the update scheme (4.1)-(4.2) by introducing a damping strategy inspired by the work in [37]. Let

$$\gamma_{j-1} = \max\left\{\frac{\mathbf{y}_{j-1}^\top \mathbf{y}_{j-1}}{\mathbf{s}_{j-1}^\top \mathbf{y}_{j-1}},\ \delta\right\} \geq \delta, \tag{4.6}$$

with $\delta > 0$. We replace $\mathbf{y}_j$ with the vector

$$\bar{\mathbf{y}}_j = \nu_j \mathbf{y}_j + (1 - \nu_j)\,\gamma_{j-1}\mathbf{s}_j, \tag{4.7}$$

where the weight ν_j is determined as follows:

$$\nu_j = \begin{cases} \frac{0.75\,\gamma_{j-1}\,\mathbf{s}_j^\top \mathbf{s}_j}{\gamma_{j-1}\,\mathbf{s}_j^\top \mathbf{s}_j - \mathbf{s}_j^\top \mathbf{y}_j}, & \text{if } \mathbf{s}_j^\top \mathbf{y}_j < 0.25\,\gamma_{j-1}\,\mathbf{s}_j^\top \mathbf{s}_j, \\ 1, & \text{otherwise.} \end{cases} \tag{4.8}$$

Note that Lemma 3.2 and Lemma 3.3 in [37] guarantee that the L-BFGS matrices defined by (4.4) and (4.6)-(4.8) satisfy (4.5), and hence Assumption 3.2.

Notice that we can replace the subsampled gradient estimate with alternative gradient estimates coming, e.g., from variance-reduction techniques, which have gained much attention in the literature. This is the case of the stochastic L-BFGS algorithm by Moritz et al. [28], the stochastic block L-BFGS by Gower et al. [17], and the stochastic damped L-BFGS method by Wang et al. [37], where SVRG gradient approximations are used. The first two methods are suited for strongly convex optimization problems. The method in [28] computes the same inverse Hessian approximation as in [8], while the method in [17] uses an adaptive sketching technique exploiting the action of a subsampled Hessian on a set of random vectors rather than just on a single vector. Both stochastic BFGS algorithms use constant step lengths and have Q-linear rate of convergence of the expected value of the error in the objective function, but the block L-BFGS one appears more efficient than the other in most of the numerical experiments reported in [17]. The method in [37], designed for nonconvex problems, uses damped L-BFGS updates, which are computed at each iteration by using a difference of gradients (the gradient sample at the previous iteration is used to ensure independence). Also in this case a constant step lenght is considered, and the authors prove that the expected value of the gradient norm (computed among all the iterations) is led below a predefined threshold in a finite number of steps.

Instead of the SVRG approximation, we apply a mini-batch variant of the SAGA algorithm [13], used in [18]. Starting from the matrix $J^0 \in \mathbb{R}^{n\times N}$ whose columns are defined as $J_0^{(i)} = \nabla\phi_i(\mathbf{x}^0)$, at each iteration we compute the gradient approximation as

$$\mathbf{g}_{\mathcal{N}_k}^{\text{SAGA}}(\mathbf{x}_k) = \frac{1}{N_k}\sum_{i\in\mathcal{N}_k}\left(\nabla\phi_i(\mathbf{x}_k) - J_k^{(i)}\right) + \frac{1}{N}\sum_{l=1}^{N} J_k^{(l)}, \tag{4.9}$$

and, after updating the iterate, we set

$$J_{k+1}^{(i)} = \begin{cases} J_k^{(i)} & \text{if } i \notin \mathcal{N}_k, \\ \nabla\phi_i(\mathbf{x}_{k+1}) & \text{if } i \in \mathcal{N}_k. \end{cases} \tag{4.10}$$

As in SVRG, the set $\mathcal{N}$ is partitioned into a fixed-number n_b of random mini-batches which are used in order. However, SAGA only requires a full gradient computation at the beginning of the algorithm, while SVRG requires a full gradient evaluation every n_b iterations. Note that the SAGA gradient estimate satisfies the first part of Assumption 3.3 (gradient unbiasedness). Furthermore, by [19, Lemma 13], it also satisfies the second part of Assumption 3.3 (variance boundedness) if the iterates are bounded.

The resulting method, named LSOS-BFGS, is reported in Algorithm 2. Note that the first L-BFGS update pair is available after the first $2l$ iterations, and following [8] we take $\mathbf{d}_k = -\mathbf{g}(\mathbf{x}_k)$ for the first $2l$ iterations.

Remark 4.1. By assuming that all the functions ϕ_i have Lipschitz-continuous gradients with Lipschitz constant bounded from above by L, we have that the gradient estimate $\mathbf{g}_{\mathcal{N}_k}^{\text{SAGA}}(\mathbf{x})$ is Lipschitz continuous with Lipschitz constant bounded from above by L.

5. Numerical experiments

We developed a MATLAB implementation of Algorithm LSOS-BFGS and tested it on both nonconvex and convex finite-sum problems arising in machine learning.

The nonconvex problems we considered are nonlinear least-squares problems as the ones in [39]. Given N pairs $(\mathbf{a}_i, b_i)$, where $\mathbf{a}_i \in \mathbb{R}^n$ is a data point and $b_i \in \{0, 1\}$ the corresponding response, and considering a sigmoidal kernel, one can obtain a

Algorithm 2 LSOS-BFGS

```
given $\mathbf{x}^0 \in \mathbb{R}^n$, $\eta, \vartheta \in (0,1)$, $\{\alpha_k\} \subset \mathbb{R}_{++}$, $c_{\min}, C_{\max} \in \mathbb{R}_+$, $K_{\max} \in \mathbb{N}$, $n_b, l, m \in \mathbb{N}$
$K_f = 0$, $ind = 0$, $k = 0$
while stop criterion not satisfied do
  compute a random and uniform partition $\{\mathcal{K}_0, \mathcal{K}_1, \ldots, \mathcal{K}_{n_b-1}\}$ of $\mathcal{N}$
  for $r = 0, \ldots, n_b - 1$ do
    choose $\mathcal{N}_k = \mathcal{K}_r$ and compute $\mathbf{g}_k = \mathbf{g}^{\mathrm{SAGA}}_{\mathcal{N}_k}(\mathbf{x}_k)$ as in (4.9)-(4.10)
    if $k < 2l$ then
      $\mathbf{d}_k = -\mathbf{g}_k$
    else
      $\mathbf{d}_k = -H_k\, \mathbf{g}_k$ with $H_k$ defined in (4.3)-(4.4)
    end if
    if  $ind = 0$ then
      find the smallest integer $j \geq 0$ such that the step length $t_k = \beta^j$ satisfies
                $f_{\mathcal{N}_k}(\mathbf{x}_k + t_k \mathbf{d}_k) \leq f_{\mathcal{N}_k}(\mathbf{x}_k) + \eta\, t_k\, \mathbf{g}_k^\top \mathbf{d}_k + \vartheta^k$
      $\overline{\mathbf{x}}_k = \mathbf{x}_k + t_k \mathbf{d}_k$
      choose $\mathcal{D}_k \subset \mathcal{N}$ randomly and uniformly
      if $f_{\mathcal{D}_k}(\overline{\mathbf{x}}_k) \leq f_{\mathcal{D}_k}(\mathbf{x}_k) - c_{\min} \|\mathbf{g}_{\mathcal{D}_k}(\mathbf{x}_k)\|^2 + C_{\max}\, \zeta_k$ then
        $\mathbf{x}_{k+1} = \overline{\mathbf{x}}_k$
      else
        $\mathbf{x}_{k+1} = \mathbf{x}_k$
        $K_f = K_f + 1$
        if $K_f > K_{\max}$ then
          $ind = 1$
        end if
      end if
    else
      $t_k = \alpha_k$
      $\mathbf{x}_{k+1} = \mathbf{x}_k + t_k \mathbf{d}_k$
    end if
    $k = k + 1$
    if mod $(k, l) = 0$ and $k \geq 2l$ then
      update the L-BFGS correction pairs by using (4.1), and (4.2) or (4.6)-(4.8)
    end if
  end for
end while
```

problem of the form (1.1), where

$$\phi_i(\mathbf{x}) = \frac{1}{2}\left(b_i - \frac{1}{1 + e^{-\mathbf{a}_i^\top \mathbf{x}}}\right)^2 .$$

By setting $u_i(\mathbf{x}) = (1 + e^{-\mathbf{a}_i^\top \mathbf{x}})^{-1}$, the gradient and the Hessian of ϕ_i can be written as

$$\begin{aligned} \nabla \phi_i(\mathbf{x}) &= -\left(u_i(\mathbf{x})\,(1 - u_i(\mathbf{x}))\,(b_i - u_i(\mathbf{x}))\right)\mathbf{a}_i, \\ \nabla^2 \phi_i(\mathbf{x}) &= -u_i(\mathbf{x})\,(1 - u_i(\mathbf{x}))\left(b_i - 2\,(1 + b_i)u_i(\mathbf{x}) + 3\,u_i(\mathbf{x})^2\right)\mathbf{a}_i\mathbf{a}_i^\top . \end{aligned}$$

The convex problems come from the training of a linear classifier by minimizing the ℓ_2-regularized logistic regression model. Given the pairs $(\mathbf{a}_i, b_i)$, where $b_i \in \{-1, 1\}$ is the class label associated with the training point $\mathbf{a}_i \in \mathbb{R}^n$, an unbiased hyperplane approximately separating the two classes can be found by solving problem (1.1), where

$$\phi_i(\mathbf{x}) = \log\left(1 + e^{-b_i\,\mathbf{a}_i^\top \mathbf{x}}\right) + \frac{\mu}{2}\|\mathbf{x}\|^2$$

and $\mu > 0$. By setting $z_i(\mathbf{x}) = 1 + e^{-b_i\,\mathbf{a}_i^\top \mathbf{x}}$, the gradient and the Hessian of ϕ_i can be written as

$$\nabla\phi_i(\mathbf{x}) = \frac{1 - z_i(\mathbf{x})}{z_i(\mathbf{x})} b_i\,\mathbf{a}_i + \mu\mathbf{x}, \qquad \nabla^2\phi_i(\mathbf{x}) = \frac{z_i(\mathbf{x}) - 1}{z_i^2(\mathbf{x})}\mathbf{a}_i\mathbf{a}_i^\top + \mu I.$$

Note that from $(z_i(\mathbf{x}) - 1)/z_i^2(\mathbf{x}) \in (0, 1)$ it follows that ϕ_i is μ-strongly convex and

$$\mu I \preceq \nabla^2\phi_i(\mathbf{x}) \preceq LI, \quad L = \mu + \max_{i=1,\dots,N} \|a_i\|^2.$$

We observe that in both cases $\nabla\phi_i(\mathbf{x})$ and $\nabla^2\phi_i(\mathbf{x})$ can be computed at low cost after $\phi_i(\mathbf{x})$ has been computed, because of the special form of the derivatives and the fact that the dominant cost is usually the scalar product in $u_i(\mathbf{x})$ and $z_i(\mathbf{x})$.

All the tests were run with MATLAB R2019b on the *magicbox* server operated by the Department of Mathematics and Physics at the University of Campania "L. Vanvitelli". It is equipped with 8 Intel Xeon Platinum 8168 CPUs, 1536 GB of RAM and Linux CentOS 7.5 operating system. A single Intel Xeon CPU with 192 GB of RAM was used in the experiments.

5.1. **Results on nonconvex problems.** To assess the performance of LSOS-BFGS on the solution of nonconvex problems, we compared it with the following algorithms:

- the variance-reduced stochastic damped L-BFGS algorithm SdLBFGS-VR [37];
- a mini-batch variant of the SAGA algorithm equipped with the same line search used in LSOS-BFGS, referred to as SAGA.

We developed our own MATLAB implementation of SdLBFGS-VR, ensuring consistency with LSOS-BFGS in terms of function and gradient evaluation costs. Since the convergence results in the nonconvex case are stated in terms of gradient norm (see Theorem 3.10 and the results in [37]), that value was used as a measure of optimality. The constant step length for SdLBFGS-VR was chosen by means of a grid search over the set $S = \{1, 5\cdot 10^{-1}, 10^{-1}, \dots, 5\cdot 10^{-4}, 10^{-4}\}$, selecting the step length that yielded the smallest gradient in a fixed execution time. Moreover, we set the L-BFGS memory equal to 10 and $\delta = 10^{-2}$ in the damping strategy. In Algorithms LSOS-BFGS and SAGA we set $\vartheta = 0.999$ and the initial line-search step length equal to 1. Concerning the L-BFGS update in LSOS-BFGS, the parameters were set as $m = 10$ and $l = 5$, and $\delta = 10^{-2}$ in (4.6). We also set, for both LSOS-BFGS and SAGA, $c_{\min} = 10^{-6}$, $C_{\max} = 10^2$ and $K_{\max} = 10^5$. Finally, we considered the following LSOS-BFGS specific quantities: $\mathcal{D}_k$ with cardinality 1, the predefined step length $\alpha_k = \frac{1}{\|\mathbf{d}_0\|}\frac{T}{T+k}$, with $T = 10^6$, the gradient sample size equal to $N_k = \lceil\sqrt{N}\rceil$ and the Hessian sample size ($|\mathcal{T}_j|$ in (4.2)) equal to $3\lceil\sqrt{N}\rceil$.

The comparison was performed by using binary classification datasets from the LIBSVM collection (`https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`). The results on the four problems listed in Table 1 are representative of the general behavior of LSOS-BFGS.

The algorithms were stopped when a maximum execution time was reached, i.e., 60 seconds for w8a and gisette, and 300 seconds for real-sim and rcv1. Each algorithm

Table 1. Datasets from LIBSVM used in the nonconvex numerical experiments and in the comparison of LSOS-BFGS with GGR, MNJ and SAGA. For each dataset the number of training points and the number of features (space dimension) are reported. Whenever a training set was not specified in LIBSVM, we selected it by using the MATLAB `crossvalind` function so that it contained 70% of the available data.

name	N	n
gisette	6000	5000
rcv1	20242	47236
real-sim	50617	20958
w8a	49749	300

was run 20 times on each problem and the average error and average execution time spent until each iteration k were computed. In the plots we represent the mean optimality measures as lines together with shaded regions corresponding to 95% confidence intervals.

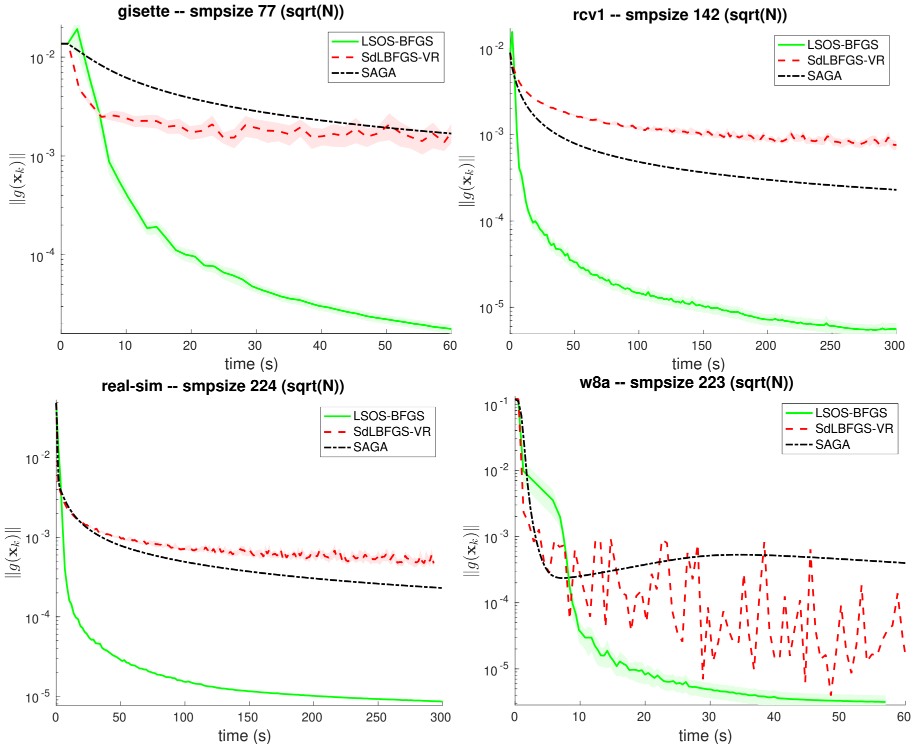

Figure 1. Nonconvex least squares problems: comparison of LSOS-BFGS, SdLBFGS-VR and SAGA in terms of gradient norm versus execution time.

Figure 1 shows a comparison among the three algorithms in terms of the average gradient norm versus the average execution time. For SdLBFGS-VR, the grid search for the step lengths was performed on the first of the 20 runs and then fixed for the remaining 19 runs. The results show that LSOS-BFGS is more effective than the two competitors in reducing the norm of the gradient of the objective function, and hence in converging to stationary points. It is worth noting that the average number of rejected steps (K_f in Algorithm LSOS-BFGS) was 0 for gisette, below 1% for w8a and real-sim, and around 6% for rcv1. The maximum number of failures ($K_{\max}$) was never reached, hence LSOS-BFGS never resorted to the use of SA step lengths.

5.2. **Results on convex problems.** We performed two different sets of experiments for assessing the performance of LSOS-BFGS on the solution of strongly convex problems. In the first one LSOS-BFGS was compared with the following algorithms:

- the stochastic L-BFGS algorithms proposed in [17], referred to as GGR;
- the stochastic L-BFGS algorithms proposed in [28], referred to as MNJ;
- the mini-batch variant of the SAGA used in the previous experiments.

The implementations of GGR and MNJ were taken from the MATLAB StochBFGS code available from `https://perso.telecom-paristech.fr/rgower/software/StochBFGS_dist-0.0.zip` and were used with constant step lengths selected by means of a grid search over the same set S used in the nonconvex case. We chose the step lengths leading to the best results in terms of objective function error versus execution time. On this class of problems we observed that the performance of LSOS-BFGS improved if the line search was started from a value smaller than 1 (which led to a reduction in the number of line-search steps performed at each iteration). Thus, we began the line search from a value t_{ini} selected by means of a grid search over S, obtaining $t_{\text{ini}} = 1 \cdot 10^{-2}$ for gisette and w8a, and $t_{\text{ini}} = 5 \cdot 10^{-3}$ for rcv1 and real-sim. The same strategy was adopted for the line-search version of SAGA, obtaining $t_{\text{ini}} = 1 \cdot 10^{-1}$ for gisette, $t_{\text{ini}} = 1$ for rcv1 and real-sim, and $t_{\text{ini}} = 5 \cdot 10^{-2}$ for w8a. The sample $\mathcal{D}_k$ at line 15 of LSOS-BFGS algorithm was again of size 1. Concerning the L-BFGS update, we set $m = 10$ and $l = 5$ as in the case of nonconvex problems, and used these values also in the MNJ algorithm. For GGR, following the indications coming from the results in [17], we set $m = 5$ and used the sketching based on the previous directions (indicated as `prev` in [17]), with sketch size $l = \lceil \sqrt[3]{n} \rceil$.

This comparison was performed by using again the datasets listed in Table 1. According to the experiments reported in [17], we set the gradient sample size equal to $\lceil \sqrt{N} \rceil$, the Hessian sample size for LSOS-BFGS and MNJ equal to $3\lceil \sqrt{N} \rceil$, and $\mu = 1/N$ as regularization parameter. The same stopping criterion (based on execution time) was used for the nonconvex problem instances built with the same datasets. For each problem we computed a solution with high accuracy by using the (deterministic) L-BFGS implementation by Mark Schmidt, available from `https://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`.

Figures 2 and 3 show a comparison among the four algorithms in terms of the average absolute error on the objective function (with respect to the optimal value) versus the average execution time and the number of data passes, respectively. As in the previous experiments, the error and the times are averaged over 20 runs and the plots show the 95% confidence interval with respect to the error. The grid

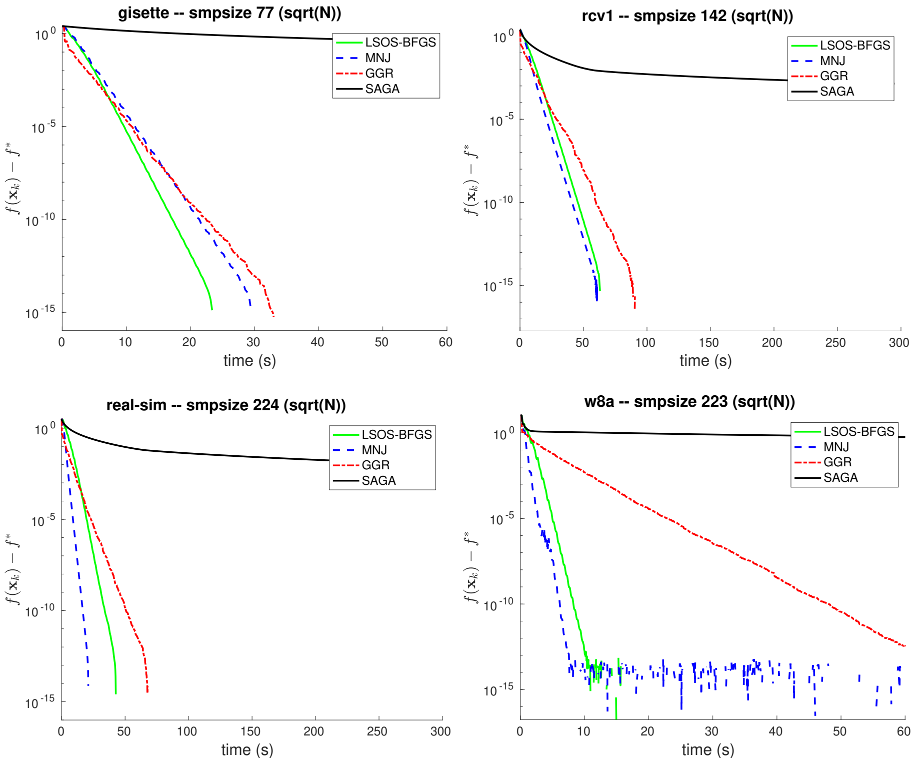

FIGURE 2. Binary classification problems: comparison of LSOS-BFGS, MNJ, GGR and SAGA in terms of function error versus execution time.

search for defining or initializing the step lengths was performed again on the first of the 20 runs and then fixed for the remaining 19 ones.

The results in terms of execution time (Figure 2) show that LSOS-BFGS outperforms the other stochastic L-BFGS algorithms on gisette, and outperforms GGR on rcv1, real-sim and w8a; furthermore, it is comparable with MNJ on rcv1. Moreover, LSOS-BFGS is always more efficient than the line-search-based mini-batch SAGA, showing that the introduction of stochastic second-order information is crucial for the performance of the algorithm.

We now focus on the comparison in terms of number of data passes, which provides a measure of oracle complexity for the four algorithms. By looking at Figure 3 it is clear how, thanks to the use of line searches and SAGA (which does not need to compute a full gradient at each step), LSOS-BFGS is able to outperform its competitors on all the cases but w8a, on which it is comparable with MNJ. We observe that in LSOS-BFGS the average number of rejected steps was 0 for gisette and below 6% for rcv1, real-sim and w8a. Again, the maximum number of failures ($K_{\max}$) was never reached, and LSOS-BFGS always determined the step lengths by the nonmonotone line search.

In the second set of experiments on convex problems we compared LSOS-BFGS with the Incremental Quasi-Newton (IQN) method [25]. Note that the latter has proven superlinear convergence rate, but high memory requirements, i.e., multiple

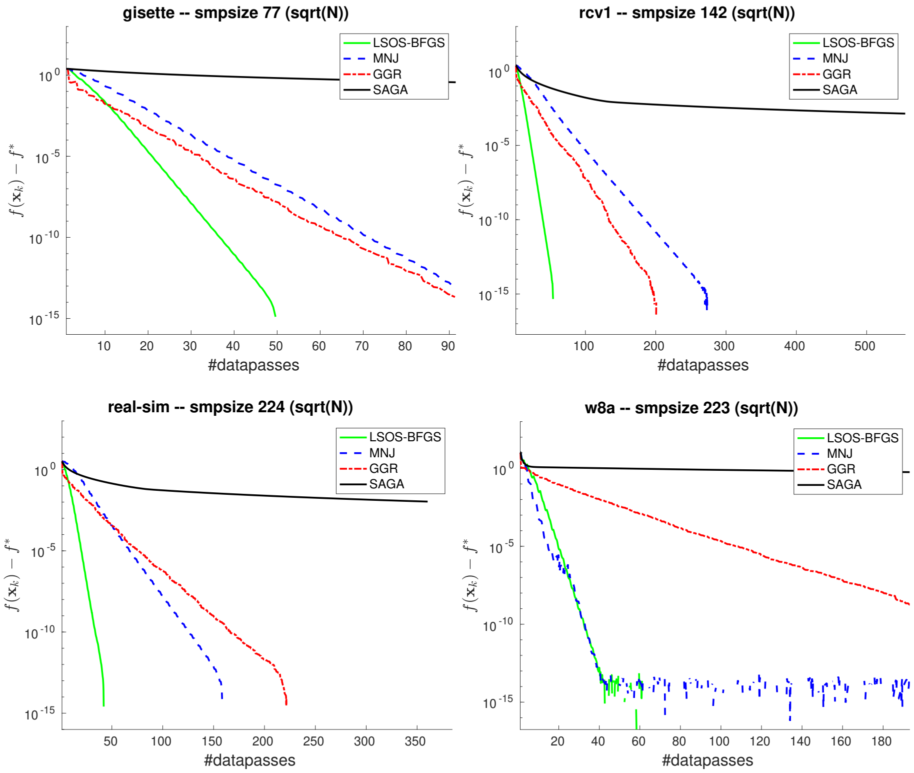

FIGURE 3. Binary classification problems: comparison of LSOS-BFGS, MNJ, GGR and SAGA in terms of function error versus data passes.

Hessian matrices to be stored. We used the MATLAB implementation of IQN available from `https://github.com/hiroyuki-kasai/SGDLibrary`, but in our opinion we improved it by avoiding some redundant operations and explicit inversions of matrices. We note that for these experiments we had to use smaller datasets (see Table 2), because of the high memory cost of the IQN implementation, which requires storing a full Hessian matrix for each sample in the training set. Thus, in LSOS-BFGS we set the gradient sample size as $N_k = 10$ and the Hessian sample size equal to $|\mathcal{T}_j| = 30$. The remaining L-BFGS parameters were chosen as in the previous experiments, including the cardinality of $\mathcal{D}_k$ equal to 1. Moreover, we set to 10^{-1} the starting value of the step length in the line search for all the four datasets. Recall that for the superlinear convergence property to hold, IQN had to be used with step length 1.

Figures 4 and 5 show a comparison between LSOS-BFGS and IQN in terms of the average absolute error on the objective function (with respect to the optimal value computed with the L-BFGS code by Mark Schmidt) versus the average execution time and the number of data passes, respectively. It is worth mentioning that for IQN a data pass corresponds exactly to an epoch. For the tests reported in Figure 4 we set the maximum execution time for the two algorithms equal to 60 seconds. For the tests reported in Figure 5, we first run LSOS-BFGS for 60 seconds and then

TABLE 2. Datasets from LIBSVM used in the comparison of LSOS-BFGS with IQN on convex problems. For each dataset the number of training points and the number of features (space dimension) are reported.

name	N	n
a9a	32561	123
cod-rna	59535	8
mushrooms	8124	112
phishing	11055	68

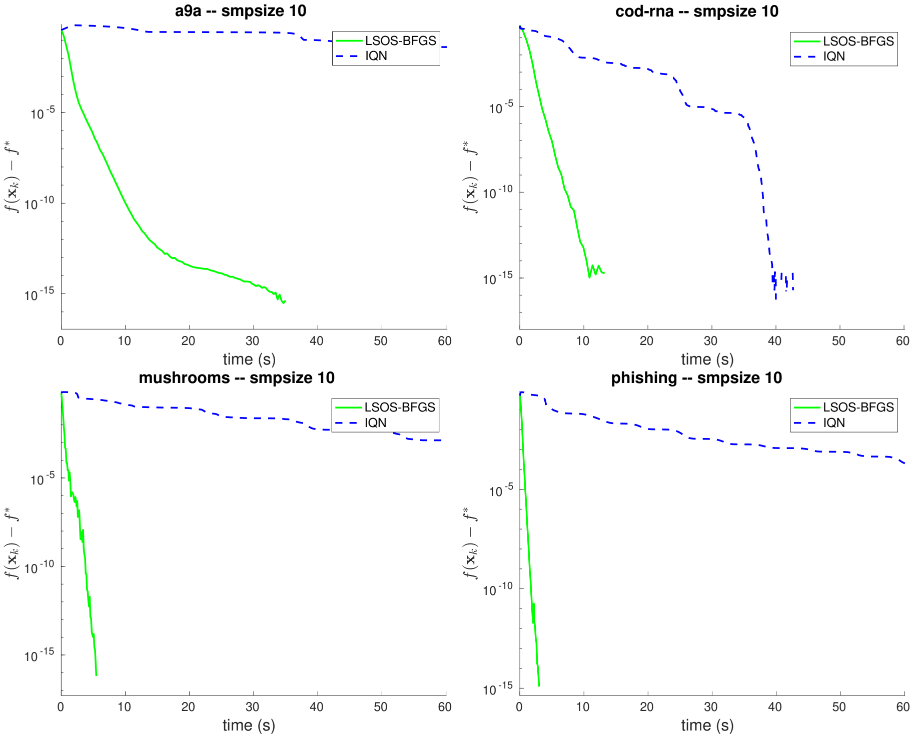

FIGURE 4. Binary classification problems: comparison of LSOS-BFGS and IQN in terms of function error versus execution time.

run IQN to perform around as many data passes as the ones completed by LSOS-BFGS in that time frame. Like the other cases, the values plotted for LSOS-BFGS are averaged over 20 runs. The shaded areas corresponding to the 95% confidence interval are almost invisible, indicating low variance in the results. Since IQN spans the sample set cyclically, picking one sample at each iteration, IQN was run once for each test.

From Figure 4 LSOS-BFGS appears more efficient in terms of execution time for all the problems. This is possibly due to the cost which has to be paid to obtain the theoretical superlinear convergence rate in IQN, which requires at each iteration the solution of a dense linear system. Interestingly, despite the superlinear convergence rate of IQN, LSOS-BFGS is also able to outperform it when the comparison is made

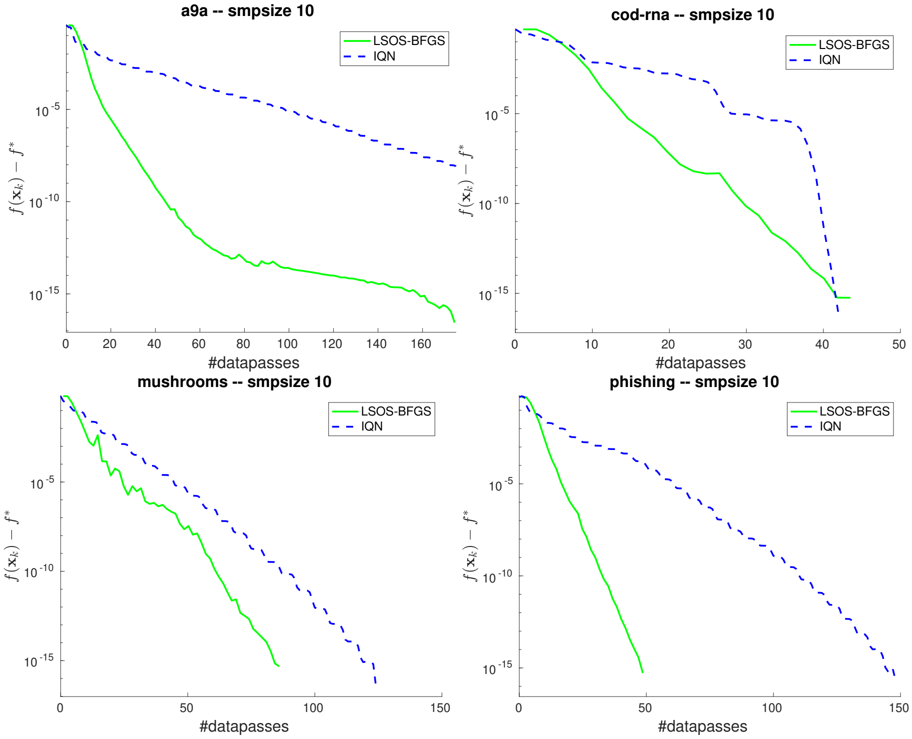

Figure 5. Binary classification problems: comparison of LSOS-BFGS and IQN in terms of function error versus data passes.

in terms of oracle complexity (Figure 5). This suggests that, although we cannot prove a superlinear rate of convergence for the algorithms fitting into the LSOS framework, this approach can yield efficient algorithms in practice.

6. Conclusions and outlook

We introduced a novel stochastic line-search algorithmic framework called LSOS, for the solution of nonconvex finite-sum problems, which allows the use of inexact second-order directions. Almost sure convergence to a stationary point for all the algorithms fitting into the LSOS framework was proved. Moreover, for strongly convex problems, we proved a.s. convergence of the sequence of iterates to the unique minimizer. Numerical experiments showed that combining stochastic L-BFGS Hessian approximations with the SAGA variance-reduction technique and with line searches produces methods that are highly competitive with state-of-the art first- and second-order stochastic optimization methods both when accounting for computational time and when accounting for oracle complexity.

Future work will be focused on the application of methods from the LSOS class to nonconvex problems arising in the training of neural networks. Moreover, we intend to investigate the extension to the stochastic setting of the strategies for combining first- and second-order directions proposed in [14, 15]. Finally, a challenging future research agenda includes the extension of (some of) the results presented in this work to constrained problems.

Acknowledgement. We are thankful to the associate editor and two anonymous referees whose comments helped us improve the paper.

References

1. Stefania Bellavia, Gianmarco Gurioli, and Benedetta Morini, *Adaptive cubic regularization methods with dynamic inexact Hessian information and applications to finite-sum minimization*, IMA Journal of Numerical Analysis **41** (2021), no. 1, 764–799. MR 4205071
2. Stefania Bellavia, Nataša Krejić, and Nataša Krklec Jerinkić, *Subsampled inexact Newton methods for minimizing large sums of convex functions*, IMA Journal of Numerical Analysis **40** (2020), no. 4, 2309–2341. MR 4167047
3. Albert S. Berahas, Liyuan Cao, and Katya Scheinberg, *Global convergence rate analysis of a generic line search algorithm with noise*, SIAM Journal on Optimization **31** (2021), no. 2, 1489–1518.
4. Albert S. Berahas and Martin Takáč, *A robust multi-batch L-BFGS method for machine learning*, Optim. Methods Softw. **35** (2020), no. 1, 191–219. MR 4032946
5. Dimitri P. Bertsekas, *Nonlinear programming*, second ed., Athena Scientific Optimization and Computation Series, Athena Scientific, Belmont, MA, 1999. MR 3444832
6. Raghu Bollapragada, Richard H. Byrd, and Jorge Nocedal, *Exact and inexact subsampled Newton methods for optimization*, IMA J. Numer. Anal. **39** (2019), no. 2, 545–578. MR 3941877
7. Léon Bottou, Frank E. Curtis, and Jorge Nocedal, *Optimization methods for large-scale machine learning*, SIAM Rev. **60** (2018), no. 2, 223–311. MR 3797719
8. R. H. Byrd, S. L. Hansen, Jorge Nocedal, and Y. Singer, *A stochastic Quasi-Newton method for large-scale optimization*, SIAM J. Optim. **26** (2016), no. 2, 1008–1031. MR 3485979
9. Richard H. Byrd, Gillian M. Chin, Will Neveitt, and Jorge Nocedal, *On the use of stochastic Hessian information in optimization methods for machine learning*, SIAM J. Optim. **21** (2011), no. 3, 977–995. MR 2837560
10. Richard H. Byrd, Gillian M. Chin, Jorge Nocedal, and Yuchen Wu, *Sample size selection in optimization methods for machine learning*, Math. Program. **134** (2012), no. 1, Ser. B, 127–155. MR 2947555
11. Huiming Chen, Ho-Chun Wu, Shing-Chow Chan, and Wong-Hing Lam, *A stochastic quasi-newton method for large-scale nonconvex optimization with applications*, IEEE Transactions on Neural Networks and Learning Systems **31** (2020), no. 11, 4776–4790.
12. Frank E. Curtis and Rui Shi, *A fully stochastic second-order trust region method*, Optimization Methods and Software **0** (2020), no. 0, 1–34.
13. Aaron Defazio, Francis Bach, and Simon Lacoste-Julien, *SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives*, Advances in Neural Information Processing Systems 27 (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), Curran Associates, Inc., 2014, pp. 1646–1654.
14. Daniela di Serafino, Gerardo Toraldo, and Marco Viola, *A Gradient-Based Globalization Strategy for the Newton Method*, In: Sergeyev Y.D. and Kvasov D.E. (eds) Numerical Computations: Theory and Algorithms (Cham), Lecture Notes in Computer Science, vol. 11973, Springer International Publishing, 2020, pp. 177–185.
15. ———, *Using gradient directions to get global convergence of Newton-type methods*, Applied Mathematics and Computation (2020), 125612.
16. Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT Press, 2016, `http://www.deeplearningbook.org`.
17. Robert M. Gower, Donald Goldfarb, and Peter Richtárik, *Stochastic block BFGS: Squeezing more curvature out of data*, Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, JMLR.org, 2016, pp. 1869–1878.
18. Robert M. Gower, Peter Richtárik, and Francis Bach, *Stochastic quasi-gradient methods: variance reduction via Jacobian sketching*, Mathematical Programming (2020).
19. Samuel Horváth and Peter Richtarik, *Nonconvex variance reduced optimization with arbitrary sampling*, Proceedings of the 36th International Conference on Machine Learning (Kamalika Chaudhuri and Ruslan Salakhutdinov, eds.), Proceedings of Machine Learning Research, vol. 97, PMLR, 09–15 Jun 2019, pp. 2781–2789.

20. Alfredo N. Iusem, Alejandro Jofré, Roberto I. Oliveira, and Philip Thompson, *Variance-based extragradient methods with line search for stochastic variational inequalities*, SIAM J. Optim. **29** (2019), no. 1, 175–206. MR 3900801
21. Majid Jahani, MohammadReza Nazari, Rachael Tappenden, Albert Berahas, and Martin Takáč, *SONIA: A symmetric blockwise truncated optimization algorithm*, Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (Arindam Banerjee and Kenji Fukumizu, eds.), Proceedings of Machine Learning Research, vol. 130, PMLR, 13–15 Apr 2021, pp. 487–495.
22. Rie Johnson and Tong Zhang, *Accelerating stochastic gradient descent using predictive variance reduction*, Advances in Neural Information Processing Systems 26 (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), Curran Associates, Inc., 2013, pp. 315–323.
23. Achim Klenke, *Probability theory*, second ed., Universitext, Springer, London, 2014, A comprehensive course. MR 3112259
24. Nataša Krejić, Zorana Lužanin, Zoran Ovcin, and Irena Stojkovska, *Descent direction method with line search for unconstrained optimization in noisy environment*, Optim. Methods Softw. **30** (2015), no. 6, 1164–1184. MR 3401091
25. Aryan Mokhtari, Mark Eisen, and Alejandro Ribeiro, *IQN: an incremental quasi-Newton method with local superlinear convergence rate*, SIAM J. Optim. **28** (2018), no. 2, 1670–1698. MR 3805550
26. Aryan Mokhtari and Alejandro Ribeiro, *RES: regularized stochastic BFGS algorithm*, IEEE Trans. Signal Process. **62** (2014), no. 23, 6089–6104. MR 3281504
27. ———, *Global convergence of online limited memory BFGS*, J. Mach. Learn. Res. **16** (2015), 3151–3181. MR 3450536
28. Philipp Moritz, Robert Nishihara, and Michael Jordan, *A linearly-convergent stochastic L-BFGS algorithm*, Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (Cadiz, Spain) (Arthur Gretton and Christian C. Robert, eds.), Proceedings of Machine Learning Research, vol. 51, PMLR, 09–11 May 2016, pp. 249–258.
29. Courtney Paquette and Katya Scheinberg, *A stochastic line search method with expected complexity analysis*, SIAM Journal on Optimization **30** (2020), no. 1, 349–376. MR 4060460
30. Seonho Park, Seung Hyun Jung, and Panos M. Pardalos, *Combining stochastic adaptive cubic regularization with negative curvature for nonconvex optimization*, Journal of Optimization Theory and Applications **184** (2020), no. 3, 953–971. MR 4061676
31. Herbert Robbins and Sutton Monro, *A stochastic approximation method*, Ann. Math. Statistics **22** (1951), 400–407. MR 42668
32. Herbert Robbins and David Siegmund, *A convergence theorem for non negative almost supermartingales and some applications*, Optimizing methods in statistics (Proc. Sympos., Ohio State Univ., Columbus, Ohio, 1971), 1971, pp. 233–257. MR 0343355
33. David Ruppert, *A Newton-Raphson version of the multivariate Robbins-Monro procedure*, Ann. Statist. **13** (1985), no. 1, 236–245. MR 773164
34. James C. Spall, *A second order stochastic approximation algorithm using only function measurements*, Proceedings of 1994 33rd IEEE Conference on Decision and Control, vol. 3, 1994, pp. 2472–2477.
35. ———, *Stochastic version of second-order (Newton-Raphson) optimization using only function measurements*, Proceedings of the 27th Conference on Winter Simulation (USA), WSC '95, IEEE Computer Society, 1995, p. 347–352.
36. ———, *Accelerated second-order stochastic optimization using only function measurements*, Proceedings of the 36th IEEE Conference on Decision and Control, vol. 2, 1997, pp. 1417–1424.
37. Xiao Wang, Shiqian Ma, Donald Goldfarb, and Wei Liu, *Stochastic quasi-Newton methods for nonconvex stochastic optimization*, SIAM Journal on Optimization **27** (2017), no. 2, 927–956. MR 3651489
38. Xiaoyu Wang, Xiao Wang, and Ya-xiang Yuan, *Stochastic proximal quasi-Newton methods for non-convex composite optimization*, Optimization Methods and Software **34** (2019), no. 5, 922–948. MR 4002760
39. Peng Xu, Fred Roosta, and Michael W. Mahoney, *Second-order optimization for non-convex machine learning: an empirical study*, Proceedings of the 2020 SIAM International Conference on Data Mining (SDM), 2020, pp. 199–207.

Department of Mathematics and Applications "Renato Caccioppoli", University of Naples Federico II, via Cintia, Monte S. Angelo, 80126 Napoli, Italy
Email address: daniela.diserafino@unina.it

Department of Mathematics and Informatics, Faculty of Science, University of Novi Sad, Trg Dositeja Obradovića 4, 21000 Novi Sad, Serbia
Email address: natasak@uns.ac.rs,natasa.krklec@dmi.uns.ac.rs

Department of Mathematics and Physics, University of Campania "Luigi Vanvitelli", viale A. Lincoln 5, 81100 Caserta, Italy
Email address: marco.viola@unicampania.it