A NOISE-TOLERANT QUASI-NEWTON ALGORITHM FOR UNCONSTRAINED OPTIMIZATION

HAO-JUN MICHAEL SHI*, YUCHEN XIE†, RICHARD BYRD‡, AND JORGE NOCEDAL§

Abstract. This paper describes an extension of the BFGS and L-BFGS methods for the minimization of a nonlinear function subject to errors. This work is motivated by applications that contain computational noise, employ low-precision arithmetic, or are subject to statistical noise. The classical BFGS and L-BFGS methods can fail in such circumstances because the updating procedure can be corrupted and the line search can behave erratically. The proposed method addresses these difficulties and ensures that the BFGS update is stable by employing a lengthening procedure that spaces out the points at which gradient differences are collected. A new line search, designed to tolerate errors, guarantees that the Armijo-Wolfe conditions are satisfied under most reasonable conditions, and works in conjunction with the lengthening procedure. The proposed methods are shown to enjoy convergence guarantees for strongly convex functions. Detailed implementations of the methods are presented, together with encouraging numerical results.

Key words. unconstrained optimization, quasi-Newton method, stochastic optimization, noisy optimization, derivative-free optimization, nonlinear optimization

AMS subject classifications. 90C30, 90C53, 90C56

1. Introduction.

Quasi-Newton methods, such as BFGS and L-BFGS, are used widely in practice because they require only first-order information and are yet able to construct useful quadratic models that make them faster and easier to use than the classical gradient method. However, in the presence of errors in the function and gradient evaluations, these methods may behave erratically. In this paper, we show how to design practical noise-tolerant versions of BFGS and L-BFGS that retain the robustness of their classical counterparts. The main challenge is to ensure that the updating process and the line search are not dominated by noise.

This paper builds upon the theoretical results of Xie et al. [29] who show that by incorporating a lengthening procedure, the BFGS method enjoys global convergence guarantees to a neighborhood of the solution for strongly convex functions. However, the algorithm proposed in [29] is not practical as it requires knowledge of the strong convexity parameter m of the objective function, which is normally not known. An overestimate of m may lead to an unstable iteration, whereas an underestimate can slow down convergence. The quasi-Newton algorithms proposed in this paper compute the lengthening parameter adaptively without the need for exogenous function information; they are designed for solving general nonlinear optimization problems and are supported by a convergence analysis for strongly convex objectives. A distinctive feature of our approach is the use of a new line search procedure that works in conjunction with the lengthening technique introduced in this paper.

*Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA. hjmshi@u.northwestern.edu. This author was supported by a grant from Facebook, and by National Science Foundation grant DMS-1620022.

†Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA. ycxie@u.northwestern.edu. This author was supported by the Office of Naval Research grant N00014-14-1-0313 P00003.

‡Department of Computer Science, University of Colorado, Boulder, CO, USA. richard@cs.colorado.edu. This author was supported by National Science Foundation grant DMS-1620070.

§Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA. j-nocedal@northwestern.edu. This author was supported by the Office of Naval Research grant N00014-14-1-0313 P00003, and by National Science Foundation grant DMS-1620022.

The problem under consideration is

$$\min_{x \in \mathbb{R}^d} \phi(x), \tag{1.1}$$

where $\phi : \mathbb{R}^d \to \mathbb{R}$ is a smooth function. This minimization must be performed while observing only inaccurate function and gradient information, i.e., by observing

$$f(x) = \phi(x) + \epsilon(x), \qquad g(x) = \nabla\phi(x) + e(x), \tag{1.2}$$

where the scalar ϵ and the vector e model the errors. We will consider the setting where the errors are bounded and the bounds are either known or estimated through an auxiliary procedure, such as `ECNoise` or pointwise sample variance estimation [23]. Specifically, we assume $|\epsilon(x)| \le \epsilon_f$ and $\|e(x)\|_2 \le \epsilon_g$ for all $x \in \mathbb{R}^d$, and that the algorithm has access to ϵ_f and ϵ_g.

Problems of this type arise in many practical applications, including when the noise is computational or adversarial. For example, in PDE-constrained optimization, the objective function often contains computational noise created by an inexact linear system solver [23], adaptive grids [1], or other internal computations. In those applications, the optimization method may not be able to control the size of the errors. In other cases, errors are due to stochastic noise, which can be caused, for example, by an intermediate Monte Carlo simulation [12]. In these cases, errors may be controllable via Monte Carlo sampling. Error in the gradient can also be inherited from noise in the function within derivative-free optimization while employing gradient approximations based on finite-differencing, interpolation, or smoothing [4, 5, 15, 16, 24, 26]. In this case, the gradient errors can be controlled by the choice of the finite-difference interval, but can only be diminished to a certain extent under the presence of function noise. We note, however, that there are applications where noise is not bounded or where the bounds ϵ_f, ϵ_g depend on x, in which case the methods proposed here cannot be directly applied.

The fact that the BFGS and L-BFGS methods can be unstable in the presence of noise is due to the nature of the BFGS updating procedure. One simple way to illustrate this is by recalling that the Hessian approximation is updated based on observed gradient differences:

$$g(x+p) - g(x) = \nabla\phi(x+p) + e(x+p) - \nabla\phi(x) - e(x), \quad p \in \mathbb{R}^d.$$

If $\|p\|$ is very small, the gradients of ϕ could cancel out leaving only noise differences. Thus, the standard BFGS method may falter even before the iterates approach the region where noise dominates. Although one could argue that the situation just described is unlikely in practice, it shows that convergence guarantees cannot be established in this case.

To provide more concrete numerical evidence for the need to bolster the BFGS method, we illustrate in Figure 1 the solution of the `ARWHEAD` problem [17] in which independent random noise uniformly distributed on $[-10^{-3}, 10^{-3}]$ is introduced to each component of the gradient. One can observe a very large increase in the condition number of the BFGS matrix that is unseen when noise is removed. This shows that the Hessian approximation is corrupted, and an examination of the run indicates that the line search gives rise to tiny steps once this has occurred. The `ARWHEAD` problem is chosen because it is easily solved yet clearly illustrates the instability of the BFGS matrix under the presence of noisy updates; we revisit this example in §5.1.

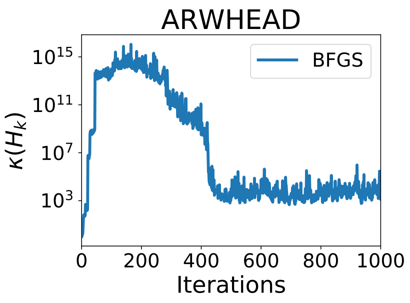

Fig. 1. *The condition number of the BFGS matrix $\kappa(H_k)$ against the number of iterations on the ARWHEAD problem with added noise.*

The literature of the BFGS method with inaccurate gradients includes the implicit filtering method of Kelley et al. [13, 20], which assumes that noise can be diminished at will at any iteration. Dennis and Walker [14] and Ypma [30] study bounded deterioration properties and local convergence of quasi-Newton methods with errors, when started near the solution with a Hessian approximation that is close to the exact Hessian. Barton [2] and Berahas, et al. [3] propose implementations of the BFGS method and L-BFGS method in which gradients are computed by an appropriate finite differencing technique, assuming that the noise level in the function evaluation is known. There has recently been some interest in designing quasi-Newton methods for machine learning applications using stochastic approximations to the gradient [7, 9, 10, 19, 25, 28]. These papers avoid potential difficulties with BFGS or L-BFGS updating by assuming that the quality of gradient differences is sufficiently controlled, and as a result, the analysis follows similar lines as for their classical counterparts. The work that is most relevant to this paper is by Xie et al. [29], who introduce the lengthening technique and establish conditions under which a steplength satisfying the Armijo-Wolfe line search conditions exists.

The contributions of this work are as follows: i) we propose practical extensions of the BFGS and L-BFGS methods for nonlinear optimization that are capable of dealing with noise by employing a new line search/lengthening technique that stabilizes the quasi-Newton update. This strategy relies on the noise control condition (2.9) introduced in this paper; ii) we provide a convergence analysis for the proposed method for strongly convex objective functions based on the properties the noise control condition instead of assuming knowledge of the strong convexity parameter, as is done in [29]; iii) we describe implementations of the methods in full detail, and present extensive numerical results that suggest that our approach is robust for certain classes of noisy optimization problems.

The paper is organized into 6 sections. In section 2, we describe the proposed algorithms, and in section 3 we establish convergence for strongly convex objectives. In section 4, we describe practical implementations of the noise-tolerant BFGS and L-BFGS methods. In section 5, we present the results of experiments on noisy synthetic examples. Lastly, we give our final remarks in section 6.

2. The Algorithm.

The BFGS and L-BFGS methods for minimizing ϕ, when only noisy observations (1.2) of the function and gradient are available, have the form

$$x_{k+1} = x_k - \alpha_k H_k g(x_k), \tag{2.1}$$

where $H_k \succ 0$ is an approximation to the inverse Hessian, $\nabla^2\phi(x_k)^{-1}$, and the steplength α_k is computed by a line search. Given a curvature pair

$$(s_k, y_k) = (x_{k+1} - x_k, g(x_{k+1}) - g(x_k)) \tag{2.2}$$
$$= (\alpha_k p_k, g(x_k + \alpha_k p_k) - g(x_k)), \tag{2.3}$$

where $p_k = -H_k g(x_k)$, the BFGS formula updates H_k as follows:

$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T, \qquad \text{where } \rho_k = 1/y_k^T s_k. \tag{2.4}$$

The L-BFGS method stores the past t curvature pairs and computes the matrix-vector product $H_k g_k$ via a two-loop recursion, with memory and computational complexity that is linear with respect to the problem dimension d [21]. For both methods, a line search ensures that $y_k^T s_k > 0$, guaranteeing that the update (2.4) is well defined.

As discussed in the previous section, the difference in gradients $g(x_k + \alpha_k p_k) - g(x_k)$ may be dominated by noise, rendering the curvature information inaccurate and potentially malign. To safeguard against this, Xie et al. [29] introduced a lengthening operation that ensures that meaningful curvature information is being collected. Specifically, they redefine the curvature pair by

$$(s_k, y_k) = (\beta_k p_k, g(x_k + \beta_k p_k) - g(x_k)), \tag{2.5}$$

where $\beta_k \geq \alpha_k$ is a sufficiently large lengthening parameter. The theoretical analysis in [29] states that setting $\beta_k = O(\epsilon_g/m\|p_k\|)$ ensures linear convergence to a neighborhood of the solution for strongly convex problems, where m is the strong convexity parameter and ϵ_g is an upper bound on the norm of the gradient noise, i.e.,

$$\|g(x) - \nabla\phi(x)\|_2 = \|e(x)\|_2 \leq \epsilon_g \qquad \forall x \in \mathbb{R}^d. \tag{2.6}$$

However, the analysis in [29] does not directly yield an implementable algorithm, as the parameter m is generally not known in practice. Furthermore, [29] does not propose a practical line search procedure for finding a steplength that satisfies the Armijo-Wolfe conditions—although it does establish the existence of such a steplength.

We now propose a rule for computing β_k that does not require knowledge of m, as well as a practical line search procedure. In our approach, we enforce the following three conditions on the steplength α_k and the lengthening parameter β_k:

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k g(x_k)^T p_k \qquad \text{(Armijo condition)} \tag{2.7}$$
$$g(x_k + \alpha_k p_k)^T p_k \geq c_2 g(x_k)^T p_k \qquad \text{(Wolfe condition)} \tag{2.8}$$
$$(g(x_k + \beta_k p_k) - g(x_k))^T p_k \geq 2(1 + c_3)\epsilon_g \|p_k\| \qquad \text{(noise control)} \tag{2.9}$$

where $0 < c_1 < c_2 < 1$ and $c_3 > 0$. Here and throughout the paper, $\|\cdot\|$ denotes the Euclidean norm. The Armijo-Wolfe conditions (2.7)–(2.8) ensure that the steplength α_k that is taken by the algorithm is not too short and yields sufficient decrease on the (noisy) objective function, while the noise control condition (2.9) on β_k is designed so that the difference in the observed directional derivatives is sufficiently large so as not to be dominated by noise. A motivation for (2.9) and a discussion of its salient properties are given below.

To satisfy the three conditions above one could find a steplength α_k that satisfies (2.7)-(2.8), and if (2.9) holds for $\beta_k = \alpha_k$, then set $\beta_k \leftarrow \alpha_k$. Otherwise, one can search for $\beta_k > \alpha_k$ to satisfy (2.9). In practice, we employ a different strategy described in section 4.1 to achieve similar objectives.

The outline of the proposed method is given in Algorithm 2.1.

Algorithm 2.1 Outline of Noise-Tolerant BFGS and L-BFGS Methods

```
Input: function $f(\cdot)$ and gradient $g(\cdot)$; noise level in gradient $\epsilon_g$; initial iterate $x_0$
    and Hessian approximation $H_0 \succ 0$;
1: for $k = 0, 1, 2, ...$ do
2:     Compute $p_k = -H_k g(x_k)$ by matrix-vector multiplication (BFGS) or two-loop
       recursion [27] (L-BFGS);
3:     Perform a line search to obtain $\alpha_k$ satisfying (2.7) and (2.8); if the line search
       fails, then compute $\alpha_k$ such that $f(x_k + \alpha_k p_k) \le f(x_k)$;
4:     Take the step $x_{k+1} = x_k + \alpha_k p_k$;
5:     Perform a lengthening procedure to obtain $\beta_k$ satisfying (2.9);
6:     Compute the curvature pair $(s_k, y_k)$ using $\beta_k$, as in (2.5);
7:     Update the Hessian approximation $H_k$ by (2.4) (BFGS) or update set $\{(s_i, y_i)\}$
       of curvature pairs (L-BFGS);
8: end for
```

The Armijo-Wolfe line search is guaranteed to find a steplength α_k that satisfies conditions (2.7)-(2.8) only when the gradient is sufficiently large relative to the noise level; otherwise p_k is not guaranteed to be a descent direction. To handle this case, a line search failure occurs when a maximum number of trial points is computed without satisfying (2.7) and (2.8). The algorithm requires an estimate of the noise level ϵ_g, which can be obtained through sampling or through the Hamming procedure described in [24]. The main remaining ingredient in this algorithm is a description of a procedure for computing α_k and β_k in step 3 and 5. This will be discussed in §4.2.

2.1. Motivation of the Noise Control Condition (2.9).

We first note that the Wolfe condition (2.8) alone does not ensure that the BFGS update is productive in the noisy setting. Even though (2.8) guarantees that

$$y_k^T s_k \ge -(1 - c_2) g(x_k)^T s_k > 0,$$

and this is sufficient for maintaining the positive definiteness of the BFGS matrix, this does not mean that y_k properly reflects the curvature of the true function, namely $\nabla\phi(x_k + \alpha_k p_k) - \nabla\phi(x_k)$, because y_k may be contaminated by noise, as discussed before.

Let us, in contrast, observe the effect of the noise control condition (2.9). We have

$$\begin{aligned}
&(g(x_k + \beta_k p_k) - g(x_k))^T p_k \\
&\qquad = [(\nabla\phi(x_k + \beta_k p_k) - \nabla\phi(x_k)) + (e(x_k + \beta_k p_k) - e(x_k))]^T p_k \\
&\qquad \le (\nabla\phi(x_k + \beta_k p_k) - \nabla\phi(x_k))^T p_k + (\|e(x_k + \beta_k p_k)\| + \|e(x_k)\|)\|p_k\| \\
&\qquad \le (\nabla\phi(x_k + \beta_k p_k) - \nabla\phi(x_k))^T p_k + 2\epsilon_g \|p_k\|,
\end{aligned}$$

by (2.6). Combining this with (2.9) we have

$$(\nabla\phi(x_k + \beta_k p_k) - \nabla\phi(x_k))^T p_k \ge 2c_3 \epsilon_g \|p_k\|, \tag{2.10}$$

and thus the true difference in the directional derivative is sufficiently large relative to the gradient noise ϵ_g. If we define

$$\tilde{y}_k = \nabla\phi(x_k + \beta_k p_k) - \nabla\phi(x_k), \tag{2.11}$$

and recall from (2.5) that $s_k = \beta_k p_k$, we can write (2.10) as

$$\tilde{y}_k^T s_k \geq 2\beta_k c_3 \epsilon_g \|p_k\|.$$

We can also establish a relationship between the observed and true curvature along the step s_k. In particular, if $\beta_k > 0$ satisfies the noise control condition (2.9) and (2.6) holds, then

$$\left|\frac{s_k^T \tilde{y}_k}{s_k^T y_k} - 1\right| \leq \frac{\|s_k\|\|\tilde{y}_k - y_k\|}{s_k^T y_k} \leq \frac{2\epsilon_g \|s_k\|}{s_k^T y_k} \leq \frac{1}{1+c_3}$$

which implies that

$$\left(1 + \frac{1}{1+c_3}\right)^{-1} s_k^T \tilde{y}_k \leq s_k^T y_k \leq \left(1 - \frac{1}{1+c_3}\right)^{-1} s_k^T \tilde{y}_k. \tag{2.12}$$

This result shows that when condition (2.9) is satisfied, the noisy curvature pair (s_k, y_k) is a good approximation to the true curvature pair $(s_k, \tilde{y}_k)$, with the parameter c_3 trading off the quality of this approximation with the locality of the curvature information being collected (in the sense that β_k may be excessively large if c_3 is chosen to be large).

Note that we are guaranteed finite termination of the lengthening procedure if there exists a $\bar{\beta} > 0$ such that for all $\beta > \bar{\beta}$,

$$\nabla\phi(x_k + \beta p_k)^T p_k \geq \nabla\phi(x_k)^T p_k + 2c_3\epsilon_g \|p_k\|.$$

This is guaranteed if $\lim_{\beta\to\infty} \nabla\phi(x_k + \beta p_k)^T p_k = \infty$, which holds for strongly convex functions, as well as for many other (but not all) functions.

Let us now verify that the noise control condition is compatible with the choice

$$\beta = O(\epsilon_g / m\|p_k\|) \tag{2.13}$$

stipulated by Xie et al. [29] in their convergence analysis of the BFGS method with errors for strongly convex functions. If ϕ is m-strongly convex, then

$$\tilde{y}_k^T p_k = (\nabla\phi(x_k + \beta_k p_k) - \nabla\phi(x_k))^T p_k \geq m\beta_k \|p_k\|^2.$$

Therefore, by our assumption, we have

$$y_k^T p_k \geq \tilde{y}_k^T p_k - 2\epsilon_g \|p_k\| \geq (m\beta_k \|p_k\| - 2\epsilon_g)\|p_k\|.$$

Therefore it suffices to ensure that

$$m\beta_k \|p_k\| - 2\epsilon_g \geq 2(1+c_3)\epsilon_g, \text{ i.e. } \beta_k \geq \frac{2(2+c_3)\epsilon_g}{m\|p_k\|}, \tag{2.14}$$

to satisfy (2.9).

Remark 1. It is natural to ask whether there are less expensive alternatives to the lengthening strategy mentioned above. The noise control condition (2.9) offers the possibility of skipping the BFGS update when it is not satisfied. We describe this approach and test it in §5. Another possibility is to use Powell damping [27, chapter 18], but we consider this to be somewhat dangerous, as it would involve repeatedly introducing spurious information in the Hessian approximation without much safeguard.

3. Convergence Analysis.

Xie et al. [29] established convergence results for the BFGS method using a lengthening strategy designed to cope with errors in the function and gradient. They assume the lengthening parameter satisfies $\beta_k\|p_k\| \geq 2\epsilon_g/m$. This leaves open the question of how to estimate the strong convexity parameter m in practice so that the convergence results in [29] still hold.

In this paper, we bypass this thorny issue and propose the lengthening strategy based on the noise control condition (2.9), which employs an estimate of the noise level of the gradient ϵ_g, but does not require knowledge of m. We now establish conditions under which Algorithm 2.1 is linearly convergent to a neighborhood of the solution determined by the noise level. We make the following assumption about the underlying function ϕ, which is standard in the analysis of quasi-Newton methods.

Assumption 3.1. *The function ϕ is m-strongly convex and has M-Lipschitz continuous gradients, i.e., there exist constants $0 < m \leq M$ such that*

$$m\|x-y\|^2 \leq \left[\nabla\phi(x) - \nabla\phi(y)\right]^T (x-y) \leq M\|x-y\|^2, \quad \forall x, y \in \mathbb{R}^d.$$

In addition, we assume that the errors in the gradient and objective function approximation are bounded.

Assumption 3.2. *There are constants $\epsilon_g \geq 0$ and $\epsilon_f \geq 0$ such that*

$$\|\nabla\phi(x) - g(x)\| \leq \epsilon_g, \ \forall x \in \mathbb{R}^d, \quad \textit{and} \tag{3.1}$$

$$|\phi(x) - f(x)| \leq \epsilon_f, \ \forall x \in \mathbb{R}^d. \tag{3.2}$$

Byrd and Nocedal [11] showed that if all curvature pairs (s_k, y_k) satisfy

$$\frac{s_k^T y_k}{s_k^T s_k} \geq \widehat{m}, \quad \frac{y_k^T y_k}{s_k^T y_k} \leq \widehat{M}, \quad \forall k \in \mathbb{N}, \tag{3.3}$$

for some constants $0 < \widehat{m} \leq \widehat{M}$, then most of the iterates generated by the (classical) BFGS method are "good iterates" in the sense that the angle between the search direction and the steepest direction is bounded away from orthogonality. This fact is used in [11] to establish convergence of the BFGS algorithm with various types of line searches for strongly convex functions.

The first step in our analysis consists of showing that bounds of the form (3.3) are satisfied for both the BFGS and L-BFGS versions of our noise tolerant Algorithm 2.1, due to the role of the noise control condition (2.9). For convenience, we summarize the notation introduced in the previous section:

$$s_k = \beta_k p_k, \quad y_k = g(x_k + s_k) - g(x_k), \quad \tilde{y}_k = \nabla\phi(x_k + s_k) - \nabla\phi(x_k),$$

and therefore the noise control condition can be written as

$$s_k^T \left[g(x_k + s_k) - g(x_k)\right] \geq c\,\epsilon_g\|s_k\|,$$

with $c = 2(1 + c_3)$.

Notation. So far we let H_k denote the BFGS approximation of the inverse Hessian. The classical analysis of the BFGS method analyzes, however, the direct Hessian approximation B_k defined as $B_k^{-1} = H_k$ [27]. Therefore, some of the results quoted from [29], are stated in terms of B_k.

LEMMA 3.3. *Suppose that Assumptions 3.1 and 3.2 hold and that $s_k \neq 0$ is chosen such that*

$$s_k^T \left[g(x_k + s_k) - g(x_k)\right] \geq c\, \epsilon_g \|s_k\|, \tag{3.4}$$

with $c > 2$ and $\epsilon_g > 0$. Then we have that

$$\frac{s_k^T y_k}{s_k^T s_k} \geq \frac{c}{c+2} m, \qquad \frac{y_k^T y_k}{s_k^T y_k} \leq \frac{c}{c-2} M. \tag{3.5}$$

Proof. Since $\|s_k\| > 0$ we have that

$$\frac{s_k^T y_k}{s_k^T s_k} \geq c \frac{\epsilon_g}{\|s_k\|} > 0.$$

In addition, since $\|\tilde{y}_k - y_k\| \leq 2\epsilon_g$ and by Assumption 3.1 we have

$$\frac{s_k^T y_k}{s_k^T s_k} \geq \frac{s_k^T \tilde{y}_k}{s_k^T s_k} - \frac{2\epsilon_g}{\|s_k\|} \geq m - \frac{2\epsilon_g}{\|s_k\|}.$$

Combining these two inequalities, we obtain

$$\frac{s_k^T y_k}{s_k^T s_k} \geq \frac{c}{c+2} m,$$

which proves the first inequality in (3.5).

For the second bound in (3.5), first note that $\|y_k\| \leq M\|s_k\| + \|\tilde{y}_k - y_k\| \leq M\|s_k\| + 2\epsilon_g$. Therefore,

$$\|s_k\| \left(M\|s_k\| + 2\epsilon_g\right) \geq \|s_k\| \|y_k\| \geq s_k^T y_k \geq c\epsilon_g \|s_k\|, \tag{3.6}$$

which yields the following lower bound on $\|s_k\|$:

$$\|s_k\| \geq (c-2)\frac{\epsilon_g}{M}. \tag{3.7}$$

Since ϕ is m-strongly convex with M-Lipschitz continuous gradients, by [8, Proposition 6.1.9 (b)] we have

$$(x-z)^T \left[\nabla\phi(x) - \nabla\phi(z)\right] \geq \frac{mM}{m+M}\|x-z\|^2 + \frac{1}{m+M}\|\nabla\phi(x) - \nabla\phi(z)\|^2, \; \forall x, z \in \mathbb{R}^d.$$

Setting $x \leftarrow x_k + s_k, z \leftarrow x_k$, and noticing that $x - z = s_k$, $\nabla\phi(x) - \nabla\phi(z) = \tilde{y}_k$, we have

$$s_k^T \tilde{y}_k \geq \frac{mM}{m+M}\|s_k\|^2 + \frac{1}{m+M}\|\tilde{y}_k\|^2.$$

By re-arranging the terms, we get

$$\|\tilde{y}_k\|^2 - (M+m)s_k^T \tilde{y}_k + \left(\frac{M+m}{2}\right)^2 \|s_k\|^2 \leq \left(\frac{M-m}{2}\right)^2 \|s_k\|^2,$$

which is equivalent to

$$\left\|\tilde{y}_k - \frac{M+m}{2} s_k\right\| \leq \frac{M-m}{2}\left\|s_k\right\|.$$

Consequently

$$\left\| y_k - \frac{M+m}{2} s_k \right\|^2 \le \left(\frac{M-m}{2} \|s_k\| + 2\epsilon_g \right)^2,$$

i.e.,

$$\begin{aligned} &\|y_k\|^2 - (M+m) s_k^T y_k + \left(\frac{M+m}{2}\right)^2 \|s_k\|^2 \\ \le &\left(\frac{M-m}{2}\right)^2 \|s_k\|^2 + 2(M-m)\|s_k\|\epsilon_g + 4\epsilon_g^2. \end{aligned}$$

Note that we have shown $s_k^T y_k > 0$, therefore, we can simplify the equation above to

$$\frac{y_k^T y_k}{s_k^T y_k} \le (M+m) + \frac{(2\epsilon_g + M\|s_k\|)(2\epsilon_g - m\|s_k\|)}{s_k^T y_k}. \tag{3.8}$$

Case 1: if $2\epsilon_g - m\|s_k\| < 0$, then we have

$$\frac{y_k^T y_k}{s_k^T y_k} \le (M+m) - \frac{\|s_k\|(2\epsilon_g + M\|s_k\|)}{s_k^T y_k} \left(m - 2\frac{\epsilon_g}{\|s_k\|} \right)$$

From (3.6) we know that

$$\|s_k\|(M\|s_k\| + 2\epsilon_g) \ge s_k^T y_k$$

therefore,

$$\frac{y_k^T y_k}{s_k^T y_k} \le M + 2\frac{\epsilon_g}{\|s_k\|}$$

Combining this with the lower bound $\|s_k\| \ge (c-2)\epsilon_g/M$ given in (3.7), we have

$$\frac{y_k^T y_k}{s_k^T y_k} \le M + \frac{2\epsilon_g}{\|s_k\|} \le M + \frac{2}{c-2}M = \frac{c}{c-2}M.$$

Case 2: if $2\epsilon_g - m\|s_k\| \ge 0$, then we have from (3.8) and (3.4)

$$\begin{aligned} \frac{y_k^T y_k}{s_k^T y_k} &\le (M+m) + \frac{(2\epsilon_g + M\|s_k\|)(2\epsilon_g - m\|s_k\|)}{c\epsilon_g\|s_k\|} \\ &= (M+m) + \frac{1}{c}\left(2 + M\frac{\|s_k\|}{\epsilon_g}\right)\left(2\frac{\epsilon_g}{\|s_k\|} - m\right). \end{aligned}$$

The right hand side increases as $\|s_k\|/\epsilon_g$ decreases, hence setting $\|s_k\|$ to the lower bound given in (3.7), we have

$$\begin{aligned} \frac{y_k^T y_k}{s_k^T y_k} &\le (M+m) + \frac{1}{c}\left(2 + M\frac{\|s_k\|}{\epsilon_g}\right)\left(2\frac{\epsilon_g}{\|s_k\|} - m\right) \\ &\le (M+m) + \frac{1}{c}\left(2 + M\frac{c-2}{M}\right)\left(2\frac{M}{c-2} - m\right) \\ &= \frac{c}{c-2}M. \end{aligned}$$

This proves the second inequality. □

As mentioned above, if we set $c = 2(1 + c_3)$ in (3.4), we obtain the noise control condition (2.9). Therefore, we have the following guarantee on the curvature pairs generated by Algorithm 2.1:

$$\frac{s_k^T y_k}{s_k^T s_k} \geq \widehat{m} = \frac{1 + c_3}{2 + c_3} m, \quad \frac{y_k^T y_k}{s_k^T y_k} \leq \widehat{M} = \left(1 + \frac{1}{c_3}\right) M, \quad k = 0, 1, 2, ... \tag{3.9}$$

To continue using the results in [11] we define, for any $\gamma > 0$, the index of "good iterates" $J(\gamma)$ as

$$J(\gamma) = \{k \in \mathbb{N} | \cos\theta_k \geq \gamma\}, \tag{3.10}$$

where $\cos\theta_k$ is the angle between $p_k = -H_k g_k$ and $-g_k$. The following lemma uses the bounds (3.9) to show that that for some values γ, the set $J(\gamma)$ contains a fraction of the iterates.

Lemma 3.4. *Let $\{x_k\}$, $\{p_k\}$ be generated by Algorithm 2.1, using either the full-BFGS or L-BFGS variant. Then for any $0 < q < 1$, there exists $\gamma > 0$ such that*

$$|J(\gamma) \cap [0, k-1]| \geq qk, \tag{3.11}$$

where $J(\gamma)$ is defined by (3.10).

Proof. For the full-BFGS variant of Algorithm 2.1, since we have shown that (3.9) holds, Theorem 2.1 in [11] guarantees that for any $0 < q < 1$, there exists $\gamma_F > 0$ such that

$$|J(\gamma_F) \cap [0, k-1]| \geq qk. \tag{3.12}$$

For the L-BFGS method with memory length t, we have $B_k = H_k^{-1} = B_{k,t}$, where $B_{k,i+1}$ are computed by applying BFGS update to $B_{k,i}$ with the curvature pair (s_{k+i-t}, y_{k+i-t}), and $B_{k,0}$ is defined by

$$B_{k,0} = \frac{1}{\gamma_k} I, \ \gamma_k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}.$$

Now we apply techniques developed in [11]. For any positive definite matrix B, let

$$\psi(B) = \operatorname{tr} B - \log\det B.$$

Since all curvature pairs $\{(s_k, y_k)\}$ satisfy (3.9), by [11, (2.9)] we have

$$\psi(B_{k,i+1}) \leq \psi(B_{k,i}) + (\widehat{M} - \log\widehat{m}).$$

Therefore, we have

$$\psi(B_k) = \psi(B_{k,t}) \leq \psi(B_{k,0}) + t(\widehat{M} - \log\widehat{m}).$$

By [11, (2.7)], we have

$$\begin{aligned} \kappa(B_k) &\leq \exp[\psi(B_k)] \leq \exp\left[\psi(B_0) + t(\widehat{M} - \log\widehat{m})\right] \\ &= \left[\gamma_k e^{1/\gamma_k}\right]^d \exp\left[t(\widehat{M} - \log\widehat{m})\right]. \end{aligned}$$

By (3.9) and the Cauchy-Schwarz inequality,

$$\widehat{m} \le \frac{s_{k-1}^T y_{k-1}}{s_{k-1}^T s_{k-1}} \le \frac{y_{k-1}^T y_{k-1}}{s_{k-1}^T y_{k-1}} = \frac{1}{\gamma_k} \le \widehat{M},$$

hence,

$$\gamma_k e^{1/\gamma_k} = e^{1/\gamma_k - \log(1/\gamma_k)} \le \exp[\widehat{M} - \log \widehat{m}],$$

which implies that

$$\kappa(B_k) \le \exp\left[(d+t)(\widehat{M} - \log \widehat{m})\right].$$

Finally, note that since $s_k = -\beta_k H_k g_k$ and $H_k B_k = I$,

$$\cos\theta_k = \frac{g_k^T H_k g_k}{\|g_k\|\|H_k g_k\|} = \frac{s_k^T B_k s_k}{\|s_k\|\|B_k s_k\|} \ge \frac{\lambda_{\min}(B_k)\|s_k\|^2}{\lambda_{\max}(B_k)\|s_k\|^2} = \frac{1}{\kappa(B_k)}$$
$$\ge \exp\left[-(d+t)(\widehat{M} - \log \widehat{m})\right].$$

Therefore, we have

$$\cos\theta_k \ge \gamma_L \equiv \exp\left[-(d+t)(\widehat{M} - \log \widehat{m})\right], \ \forall k \in \mathbb{N},$$

i.e.,

$$|J(\gamma_L) \cap [0, k-1]| = k, \ \forall k \in \mathbb{N}$$

which finishes the proof. ∎

By the discussions above, for both full-BFGS and L-BFGS variants of Algorithm 2.1, we can choose a fixed $q^* \in (0,1)$ and find $\gamma^* > 0$ such that

$$|J(\gamma^*) \cap [0, k-1]| \ge q^* k, \quad \forall k \in \mathbb{N}; \tag{3.13}$$

i.e., such that a fraction of iterates are guaranteed to be good iterates. From now on, let us fix the choice q^* and γ^*. Using the above results together with the analysis in [29] we arrive at the following convergence result.

THEOREM 3.5. *Suppose that Assumptions 3.1 and 3.2 hold. Let $\{x_k\}$ be generated by Algorithm 2.1, using either L-BFGS or standard BFGS. Fix $q^* \in (0,1)$ and choose $\gamma^* > 0$ such that (3.13) holds. Define*

$$\mathcal{N}_1 = \left\{ x \;\middle|\; \|\nabla\phi(x)\| \le \max\left\{ A\frac{\sqrt{M\epsilon_f}}{\gamma^*}, B\frac{\epsilon_g}{\gamma^*} \right\} \right\}, \tag{3.14}$$

and

$$\mathcal{N}_2 = \left\{ x \;\middle|\; \phi(x) \le 2\epsilon_f + \max_{y\in\mathcal{N}_1} \phi(y) \right\} \supseteq \mathcal{N}_1, \tag{3.15}$$

where

$$A = \max\left\{ \frac{16\sqrt{2}}{\sqrt{(c_2 - c_1)(4 - c_1 - 3c_2)}}, \frac{8}{\sqrt{c_1(1 - c_2)}} \right\}$$
$$B = \max\left\{ \frac{8}{1 - c_2}, \frac{8(1 + c_1)}{c_2 - c_1} + 6 \right\}.$$

Let

$$K = \min_k \ \{k \in \mathbb{N} \mid x_k \in \mathcal{N}_1\} \tag{3.16}$$

be the index of the first iterate that enters $\mathcal{N}_1$. Assume that for all iterates $k \in J(\gamma^*)$ such that $x_k \notin \mathcal{N}_1$ the line search procedure finds α_k satisfying (2.7)–(2.8). Then there exists $\rho \in (0,1)$ such that

$$\phi(x_k) - \phi^* \leq \rho^k \ (\phi(x_0) - \phi^*) + 2\epsilon_f, \quad \forall k \leq K-1.$$

Moreover, we have that $K < +\infty$ and

$$x_k \in \mathcal{N}_2, \quad \forall k \geq K.$$

Proof. Note that Algorithm 2.1 differs from Algorithm 2.1 of [29], only in the quasi-Newton updating strategy and lengthening procedure. This implies that the results through Theorem 3.5 of [29] concerning the existence of an Armijo-Wolfe step-size, also apply to Algorithm 2.1 of this paper, since the proofs of these these results do not depend on the update used. In Lemma 3.3 of this paper we showed that the lengthening procedure in step 5 of Algorithm 2.1 guarantees bounds on $(s_k^T y_k / s_k^T s_k)$ and $(y_k^T y_k / s_k^T y_k)$ such as those of Lemma 3.8 of [29]. Using these bounds we established Lemma 3.4 whose results are identical to those of Corollary 3.10 in [29], with γ^* replacing β_1. With that change, the rest of the results of [29], including Theorems 3.16–3.18, hold for Algorithm 2.1 of this paper, proving the theorem. ∎

Theorem 3.5 states that the iterates generated by Algorithm 2.1 converge linearly to a neighborhood of the solution $\mathcal{N}_1$, whose size depends on the noise levels ϵ_f, ϵ_g; the iterates will enter $\mathcal{N}_1$ in finite number of iterations, and will remain in a larger neighborhood $\mathcal{N}_2$ thereafter.

4. A Practical Algorithm.

In order to implement Algorithm 2.1, we need to design a practical procedure for computing the steplength α_k and the lengthening parameter β_k. This can be done in various ways, and in this section we present a technique that has performed well in practice. After describing this algorithm in detail, we present several heuristics designed to improve its practical performance.

4.1. Two-Phase Line Search and Lengthening Procedure.

Algorithm 2.1 and the convergence analysis of the previous section require that α_k and β_k satisfy conditions (2.7), (2.8) and (2.9). We now propose a procedure for computing these quantities.

The line search operates in two phases. The *initial phase* attempts to satisfy three conditions with the same parameter $\alpha_k = \beta_k$:

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k g(x_k)^T p_k \tag{4.1}$$

$$g(x_k + \alpha_k p_k)^T p_k \geq c_2 g(x_k)^T p_k \tag{4.2}$$

$$|(g(x_k + \alpha_k p_k) - g(x_k))^T p_k| \geq 2(1 + c_3)\epsilon_g \|p_k\|, \tag{4.3}$$

where $0 < c_1 < c_2 < 1$ and $c_3 > 0$. Observe that (4.3) and the Wolfe condition (4.2) imply the noise control condition (2.9) employed so far in the paper. We incorporate the absolute value in (4.3) in order to introduce a symmetric noise condition that can be used to determine when to adapt α_k and β_k independently. If $\epsilon_f = \epsilon_g = 0$, then we can guarantee that the initial phase will reduce to the standard Armijo-Wolfe line search, as we describe below.

The initial phase is done using the logic of the standard bisection search: backtracking if the Armijo condition is not satisfied, and advancing if the Armijo condition is satisfied and the Wolfe condition is not, but with one important modification. If the Armijo condition (4.1) is satisfied, we will check (4.3) prior to checking the Wolfe condition (4.2).

If at *any* iteration of the line search the noise control condition (4.3) is not satisfied or if the line search has performed more than the allowed number (N_{split}) of iterations, then the initial phase is terminated and the second phase, which we call the *split phase*, is triggered. In this phase, α_k and β_k are updated independently from each other. The steplength α_k is updated via the standard Armijo backtracking line search while the lengthening parameter β_k is lengthened independently until the conditions

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k g(x_k)^T p_k \tag{4.4}$$

$$(g(x_k + \beta_k p_k) - g(x_k))^T p_k \geq 2(1 + c_3)\epsilon_g \|p_k\| \tag{4.5}$$

are satisfied. We backtrack more aggressively (by a factor of 10) in the split phase in order to mitigate the cost of additional function evaluations. The limit N_{split} is imposed to prevent the line search from being fooled from noise indefinitely.

The two-phase line search (without heuristics) is presented in Algorithms 4.1 and 4.2. For completeness, we also present the pseudocode for the complete practical algorithm in 4.3.

By the design of the two-phase line search, our algorithm behaves the same as the standard (L-)BFGS algorithm (without interpolation) for non-noisy problems as long as N_{split} is sufficiently large because the split phase will never occur. In particular, if $\epsilon_g = 0$, then condition 4.3 will always be satisfied by any α_k and therefore the initial phase reduces to the standard Armijo-Wolfe line search. However, unlike the deterministic setting, the two-phase line search may not be guaranteed to find α_k and β_k under certain scenarios. When the iteration has reached the region where errors are large relative to the gradient, the backtracking line search may fail to find α_k; this is to be expected. A more subtle case is when the function is exceedingly flat along the search direction p_k so that even for a large β the function exhibits insufficient change in curvature; in this case the lengthening procedure may fail to find an appropriate β_k. To safeguard against both of these cases, the algorithm will terminate if it reaches a maximum number of line search iterations.

Remark 2. The two-phase algorithm just described may seem too complex. Let us consider some simpler alternative strategies. One approach is to employ only the split phase: (1) Compute α_k solely through a backtracking line search until the Armijo condition is satisfied; and (2) Computing β_k through a lengthening procedure that enforces both of the modified noise control and Wolfe conditions. However, the Wolfe condition on the steplength α_k allows the algorithm to take longer steps that may yield larger reductions in the objective function. This is in agreement with our computational experience.

A second alternative, given in Algorithm 2.1, is the approach employed by Xie et al. [29], who first solve for a steplength α_k that satisfies the Armijo-Wolfe conditions (4.1)-(4.2), then lengthen $\beta_k \geq \alpha_k$ until β_k satisfies the noise control condition (4.3). However, we have found experimentally that performing an Armijo-Wolfe line search attempting to find a steplength that satisfies the Armijo-Wolfe conditions in the presence of noise can be costly in terms of function and gradient evaluations because the

Algorithm 4.1 Two-Phase Armijo-Wolfe Line Search and Lengthening: Initial Phase

```
Input: functions $f(\cdot)$ and $g(\cdot)$; noise level $\epsilon_g$; current iterate $x$; search direction $p$;
    initial steplength $\alpha = 1$; constants $0 < c_1 < c_2 < 1$, $c_3 > 0$; maximum number of
    line search iterations before split $N_{\text{split}}$
1: $l \leftarrow 0$, $u \leftarrow \infty$;                                   ▷ Initialize brackets for bisection
2: for $i = 0, 1, 2, ..., N_{\text{split}} - 1$ do
3:     if $f(x + \alpha p) > f(x) + c_1 \alpha g(x)^T p$ then                  ▷ Armijo condition fails
4:         $u \leftarrow \alpha$;
5:         $\alpha \leftarrow (u + l)/2$;                                      ▷ Backtrack
6:     else if $|(g(x + \alpha p) - g(x))^T p| < 2(1 + c_3)\epsilon_g \|p\|$ then   ▷ Noise control
    condition fails
7:         Break (for loop)
8:     else if $g(x + \alpha p)^T p < c_2 g(x)^T p$ then                       ▷ Wolfe condition fails
9:         $l \leftarrow \alpha$;
10:        if $u = \infty$ then                                              ▷ Advance
11:            $\alpha \leftarrow 2\alpha$;
12:        else
13:            $\alpha \leftarrow (u + l)/2$;
14:        end if
15:    else                                                                  ▷ Satisfies all conditions
16:        $\beta \leftarrow \alpha$ ;
17:        Return $\alpha, \beta$;
18:    end if
19: end for
20: $\alpha, \beta \leftarrow$ SplitPhase$(f, g, \epsilon_g, x, p, \alpha, \beta)$;                 ▷ Enter split phase
21: Return $\alpha, \beta$;
```

Algorithm 4.2 Split Phase

```
Input: functions $f(\cdot)$ and $g(\cdot)$; noise level $\epsilon_g$; current iterate $x$; search direction $p$;
    initial steplength $\alpha$; initial lengthening parameter $\beta$, constants $0 < c_1 < c_2 < 1$,
    $c_3 > 0$
1: while $f(x + \alpha p) > f(x) + c_1 \alpha g(x)^T p$ do                         ▷ Armijo condition
2:     $\alpha = \alpha/10$;                                                   ▷ Backtrack
3: end while
4: while $(g(x + \beta p) - g(x))^T p < 2(1 + c_3)\epsilon_g \|p\|$ do          ▷ Noise control condition
5:     $\beta = 2\beta$;                                                       ▷ Lengthen
6: end while
7: Return $\alpha, \beta$;
```

Armijo-Wolfe line search may be fooled for many iterations in the presence of moderate to large noise relative to the gradient. In particular, enforcing the Wolfe condition on the steplength when the gradient is dominated by noise may lead to ill-advised or unnecessary changes to the steplength. Rather than doing this, we opt to split the computations of β and α earlier, as done in Algorithm 4.1 using (4.3) as a means to detect when to split and consider the Wolfe condition unreliable.

4.2. Heuristics. We now describe some heuristics that have improved the performance of the two-phase line search for the models of noise employed in our exper-

```
Algorithm 4.3 Complete Practical Noise-Tolerant BFGS and L-BFGS Methods
Input: function $f(\cdot)$ and gradient $g(\cdot)$; noise level in function $\epsilon_f$, noise level in gradient $\epsilon_g$; initial iterate $x_0$ and Hessian approximation $H_0 \succ 0$;
1: for $k = 0, 1, 2, ...$ do
2:     Compute $p_k = -H_k g(x_k)$ by matrix-vector multiplication (BFGS) or two-loop recursion [27] (L-BFGS);
3:     Perform two-phase Armijo-Wolfe line search (Algorithms 4.1 and 4.2) to find $\alpha_k$ and $\beta_k$;
4:     if $\alpha_k$ satisfies (4.1) then
5:         Compute $x_{k+1} = x_k + \alpha_k p_k$;
6:     end if
7:     if $\beta_k$ satisfies (2.9) then
8:         Compute curvature pair $(s_k, y_k) = (\beta_k p_k, g(x_k + \beta_k p_k) - g(x_k))$;
9:         Update $H_k$ by (2.4) (BFGS) or update set $\{(s_i, y_i)\}$ of curvature pairs (L-BFGS);
10:    end if
11: end for
```

iments.

I. Relaxation of Armijo Condition. The last term in the Armijo condition (4.1) ensures sufficient descent, but is useful only if the quantities involved are reliable; otherwise it is best to dispense with this term. To see this, consider the term $g(x_k)^T p_k$. Although $g(x_k)^T p_k = -g(x_k)^T H_k g(x_k) < 0$ since H_k is positive definite, this quantity could still be dominated by noise. If $g(x_k)^T p_k < -\epsilon_g \|p_k\|$, we can guarantee that $\nabla \phi(x_k)^T p_k < 0$, ensuring that p_k is a descent direction with respect to the true objective function. If instead we have that $g(x_k)^T p_k \geq -\epsilon_g \|p_k\|$, it is not guaranteed that we can make progress on the true objective function along p_k. In this case, we will consider the gradient estimate unreliable and dispense the sufficient decrease term, instead relaxing the condition to only enforce simple decrease $f(x_k + \alpha p_k) < f(x_k)$.

Another feature that is useful when the algorithm reaches a region where the noise in the function is large relative to the objective function is to relax the Armijo condition (4.1) by adding $2\epsilon_f$ to the right hand side. This relaxation will be done only after the first attempt at satisfying the standard Armijo condition fails. If p_k is a descent direction with respect to ϕ, which is ensured when $g(x_k)^T p_k < -\epsilon_g \|p_k\|$, then this relaxation guarantees finite termination of the line search component in the split phase. Other related line searches employing this relaxation of the Armijo condition have been analyzed in [6].

Combining the two strategies described above, our relaxed Armijo condition can be summarized as follows:

$$f(x_k + \alpha_k^i p_k) \begin{cases} \leq f(x_k) + c_1 \alpha_k^i g(x_k)^T p_k & \text{if } i = 0,\ g(x_k)^T p_k < -\epsilon_g \|p_k\| \\ \leq f(x_k) + c_1 \alpha_k^i g(x_k)^T p_k + 2\epsilon_f & \text{if } i \geq 1,\ g(x_k)^T p_k < -\epsilon_g \|p_k\| \\ < f(x_k) & \text{if } i = 0,\ g(x_k)^T p_k \geq -\epsilon_g \|p_k\| \\ < f(x_k) + 2\epsilon_f & \text{if } i \geq 1,\ g(x_k)^T p_k \geq -\epsilon_g \|p_k\| \end{cases} \tag{4.6}$$

where α_k^i denotes the i-th trial steplength at iteration k.

II. Reusing Previously Computed α. Over the course of the initial phase, we will track the best steplength that we have seen that satisfies the Armijo condition

$$\alpha_k^{\text{best}} \in \arg\min_{\alpha_k^i} \{ f(x_k + \alpha_k^i p_k) : (4.6) \text{ is satisfied} \} \tag{4.7}$$

as well as its corresponding function value. If the split phase is triggered, we will accept the previously computed value of $\alpha_k = \alpha_k^{\text{best}}$ that most decreased the objective function.

III. Initial Value of β. It is important to employ a good initial estimate of β when entering the split phase, in order to mitigate the cost of the search procedure. Recall from (2.14) that an appropriate value of the lengthening parameter is, roughly,

$$\beta_k = \frac{2(1+c_3)\epsilon_g}{m\|p_k\|_2}. \tag{4.8}$$

This formula relies on the strong convexity parameter m, which is generally not known, but since we are only using it to compute an initial value for β, it is not critical to estimate m accurately. In this vein, we compute a local estimate of m using the observed (s, y) pairs from prior iterations. For β_j with $j < k$ that satisfies both (4.2) and (4.3), we first compute an estimate of the curvature along the search direction p_j corresponding to the interval length β_j:

$$\bar{\mu}_j = \frac{(g(x_j + \beta_j p_j) - g(x_j))^T p_j}{\beta_j \|p_j\|_2^2}. \tag{4.9}$$

To estimate the strong convexity parameter m we track the last h values of the $\bar{\mu}$'s, then use the smallest of these:

$$\mu_k = \min\{\bar{\mu}_{k-1}, \bar{\mu}_{k-2}, ..., \bar{\mu}_{k-h}\}.$$

This aims to be only a local strong convexity estimate, whereas taking the minimum over all previous $\bar{\mu}$'s may be overly pessimistic. Let us denote by $\bar{\beta}_k$ the value obtained by making the substitution $m \leftarrow \mu_k$ in (4.8), and let β_k^i denote the ith trial lengthening parameter at iteration k, we define the initial value of the lengthening parameter for the split phase as

$$\beta_k^{i+1} = \max\{2\beta_k^i, \bar{\beta}_k\}. \tag{4.10}$$

We have observed in our tests that this procedures allows us to significantly mitigate the cost of additional gradient evaluations that are incurred when lengthening β_k, only requiring an additional $1 - 3$ gradient evaluations for the lengthening procedure in our experiments.

5. Numerical Experiments.

In this section, we present numerical results illustrating the behavior of the methods proposed in this paper on noisy optimization problems. We compare the classical methods, BFGS and L-BFGS, with their extensions, which we denote as BFGS-E and L-BFGS-E.

In addition, we study another approach suggested by the noise control condition (2.9), based on the well known strategy of skipping a quasi-Newton update when it may not be reliable. In the BFGS (Skips) and L-BFGS (Skips) methods, the quasi-Newton update is not performed if the noise control condition is not satisfied for $c_3 = 0$, that is,

$$(g(x_k + \alpha_k p_k) - g(x_k))^T p_k < 2\epsilon_g \|p_k\|. \tag{5.1}$$

Specifically, these methods compute a steplength α_k satisfying the Armijo-Wolfe conditions (2.7)-(2.8), and if condition (5.1) holds, the BFGS update is not performed and the next step is computed using the Hessian approximation B_k from the previous iteration; otherwise the iteration is identical to that of the BFGS and L-BFGS methods. (In the L-BFGS (Skips) method, the correction pair (s_k, y_k) is not stored when (5.1) holds.)

In summary, the 6 methods tested are:

1. `BFGS`: the standard BFGS method given by (2.1), (2.4);
2. `L-BFGS`: the standard L-BFGS method with memory $t = 10$; [27, chapter 7];
3. `BFGS (Skips)`: the standard BFGS method given by (2.1), (2.4), but skipping the BFGS update when (2.9) is not satisfied for $\beta_k = \alpha_k$ with $c_3 = 0$;
4. `L-BFGS (Skips)`: the standard L-BFGS method with memory $t = 10$, but skipping the L-BFGS update when (2.9) is not satisfied for $\beta_k = \alpha_k$ with $c_3 = 0$;
5. `BFGS-E`: the noise tolerant BFGS method given by Algorithms 2.1, 4.1 and 4.2;
6. `L-BFGS-E`: the noise tolerant L-BFGS method, which is identical to BFGS-E, except that the Hessian approximation is a limited memory matrix with memory $t = 10$.

The first four methods employ an Armijo-Wolfe line search that computes a steplength satisfying (2.7)-(2.8). The last two methods use the specialized line search described in Algorithms 4.1 and 4.2. In the deterministic case, it is common to employ cubic or quadratic interpolation to accelerate the Armijo-Wolfe search. We did not do so in the methods listed above, which use a simple bisection, because it is more robust in the presence of noise. The parameters for the line search and termination criteria are provided in Table 1.

Table 1
Parameter Settings for the Methods Tested

c_1	c_2	c_3	t	N_{split}
10^{-4}	0.9	0.5	10	30

We selected 41 unconstrained problems from the CUTEst collection [18] (see Table 2), and added stochastic uniform noise with different noise levels. The objective function and gradient have the form

$$f(x) = \phi(x) + \epsilon(x), \qquad g(x) = \nabla\phi(x) + e(x),$$

where we sample $\epsilon(x)$ and $[e(x)]_i$ independently with distribution

$$\epsilon(x) \sim U(-\xi_f, \xi_f), \qquad [e(x)]_i \sim U\left(-\xi_g, \xi_g\right) \text{ for } i = 1, ..., d.$$

This gives the noise bounds $|\epsilon(x)| \leq \epsilon_f = \xi_f$ and $\|e(x)\| \leq \epsilon_g = \sqrt{d}\xi_g$. Among methods for uncertainty quantification, ECNoise [23], point-wise sampling, and domain knowledge could be applied to obtain these bounds in practice. The optimal value ϕ^* for each function was obtained by applying the BFGS method to the original deterministic problem until it could not make further progress.

The performance of the methods is best understood by studying the runs on each of the 41 test problems. Since this is impractical due to space limitations, for every experiment, we selected a problem that illustrates typical behavior over the whole test set.

TABLE 2
Unconstrained CUTEst Problems Tested. d is the number of variables.

PROBLEM	d	PROBLEM	d	PROBLEM	d
ARWHEAD	100	DIXMAANL	90	MOREBV	100
BDQRTIC	100	DIXMAANM	90	NCB20B	100
CRAGGLVY	100	DIXMAANN	90	NONDIA	100
DIXMAANA	90	DIXMAANO	90	NONDQUAR	100
DIXMAANB	90	DIXMAANP	90	PENALTY1	100
DIXMAANC	90	DQDRTIC	100	QUARTC	100
DIXMAAND	90	DQRTIC	100	SINQUAD	100
DIXMAANE	90	EIGENALS	110	SPARSQUR	100
DIXMAANF	90	EIGENBLS	110	TOINTGSS	100
DIXMAANG	90	EIGENCLS	30	TQUARTIC	100
DIXMAANH	90	ENGVAL1	100	TRIDIA	100
DIXMAANI	90	FLETCBV3	100	WATSON	31
DIXMAANJ	90	FREUROTH	100	WOODS	100
DIXMAANK	90	GENROSE	100		

5.1. Experiments with Uniform Noise in the Gradient. In the first set of experiments, the gradient contains uniform noise but the function does not, i.e., $\epsilon_g > 0$ and $\epsilon_f = 0$. This allows us to test the efficiency of the lengthening procedure in a benign setting that avoids the effects of the noisy line search. In these experiments, all algorithms were run for a fixed number of iterations.

We begin by revisiting the `ARWHEAD` problem from Figure 1, where noise was inserted with $\xi_f = 0$ and $\xi_g = 10^{-3}$. The condition number of the BFGS and BFGS-E matrices is compared in Figure 2, and shows that the noise control condition (2.9) stabilizes the quasi-Newton update. It may seem surprising that in Figure 2 the con-

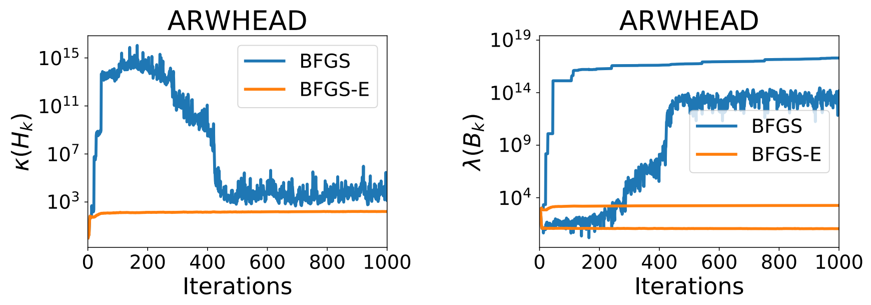

FIG. 2. *The condition number of the BFGS and BFGS-E matrices $\kappa(H_k)$ against the number of iterations (left) and the smallest and largest eigenvalues of B_k against the number of iterations (right) on the `ARWHEAD` problem. The final norm of the true gradient achieved by BFGS is approximately* 1.97e−04.

dition number of the BFGS matrix, $\kappa(H_k)$, decreases after having increased sharply. This can be explained by noting that as the iterates enter into the noisy regime, the difference in the gradient y_k can be corrupted by noise, and we may have $s_k^T y_k \gg s_k^T \tilde{y}_k$. Thus, some of the eigenvalues of the BFGS matrix $B_k = H_k^{-1}$ will increase. As the iteration proceeds, the rest of the eigenvalues become large too, hence decreasing the condition number.

Figure 3 plots the optimality gap $\phi(x) - \phi^*$ vs the number of gradient evaluations performed for the four methods on the `ARWHEAD` problem. BFGS and L-BFGS do not achieve as high accuracy in the solution as their noise-tolerant counterparts because the deterioration in the Hessian approximation leads, at some point, to the generation of very small steps that severely limit the decrease in the objective function. The behavior of the methods on this problem is typical of what we have observed. In particular BFGS-E and L-BFGS-E trigger lengthening of the curvature pairs prior to the point where BFGS and L-BFGS stagnate due to noise. This indicates that the lengthening procedure stabilizes the Hessian approximation prior to reaching this neighborhood.

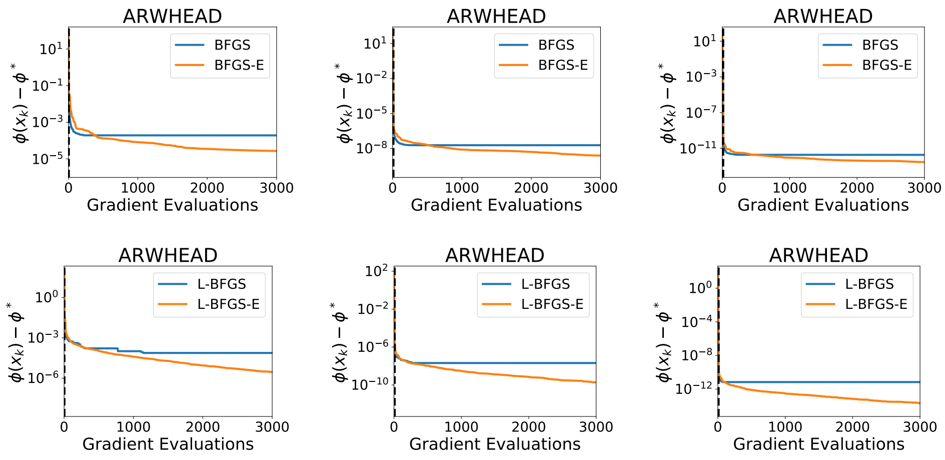

FIG. 3. *The true optimality gap $\phi(x_k) - \phi^*$ against the number of gradient evaluations on the* `ARWHEAD` *problem for $\epsilon_f = 0$, and for the following gradient noise levels: $\xi_g = 10^{-1}$ (left), 10^{-3} (middle), and 10^{-5} (right). The black dashed line denotes the iteration before the split phase becomes active.*

The lengthening procedure in our noise-tolerant algorithms comes at an additional computational cost. Figure 4 plots the cumulative number of gradient evaluations against the iteration count for the `ARWHEAD` problem. We observe that for BFGS or L-BFGS, the cumulative number of gradient evaluations is approximately equal to the number of iterations. For the noise-tolerant methods, the number of gradient evaluations match the standard BFGS and L-BFGS methods until the split phase activates. Upon entering the split phase, we notice that the cost of each iteration is approximately $2 - 4$ gradient evaluations. This may be explained by the additional $1 - 3$ gradient evaluations necessary to find the appropriate β_k that satisfies both the noise and Wolfe conditions, plus one gradient evaluation for triggering the split phase. This cost is worthwhile in that it allows the algorithm to make progress in the noisy regime.

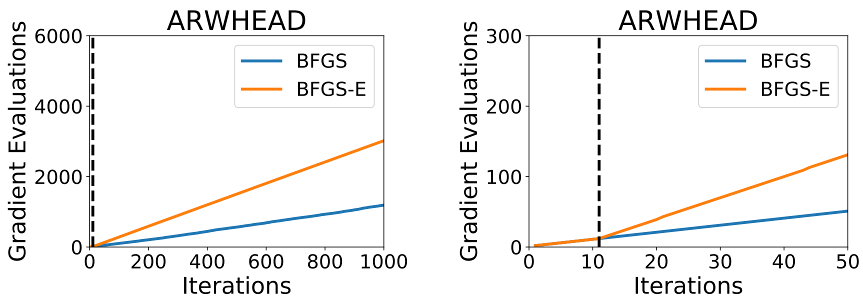

FIG. 4. *Cumulative number of gradient evaluations against the iteration count on the* `ARWHEAD` *problem for $\epsilon_f = 0$ and $\xi_g = 10^{-3}$ for BFGS and BFGS-E. The left figure plots the long-term behavior and the right figure plots the short-term behavior. The results for L-BFGS and L-BFGS-E as well as different noise levels are similar. The black dashed line denotes the iteration before the split phase becomes active.*

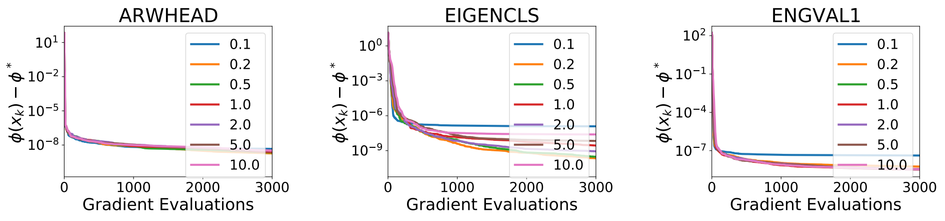

FIG. 5. *The true optimality gap $\phi(x_k) - \phi^*$ against the number of gradient evaluations applying BFGS-E on the* `ARWHEAD`, `EIGENCLS`, *and* `ENGVAL1` *problems for $\epsilon_f = 0$ and $\xi_g = 10^{-3}$ with incorrectly input $\bar{\epsilon}_g = \omega\epsilon_g$ for $\omega \in \{\frac{1}{10}, \frac{1}{5}, \frac{1}{2}, 1, 2, 5, 10\}$.*

5.1.1. Sensitivity with respect to ϵ_g. Since the bound on the gradient error ϵ_g may be estimated by an external procedure, it is possible for ϵ_g to be input incorrectly. In order to investigate the sensitivity of the choice of ϵ_g, we consider both under- and overestimation of it. We perform the same experiment with a fixed $\xi_g = 10^{-3}$ and $\epsilon_f = 0$ but provide the algorithm an incorrect $\bar{\epsilon}_g = \omega\epsilon_g$ where $\omega \in \{\frac{1}{10}, \frac{1}{5}, \frac{1}{2}, 1, 2, 5, 10\}$. This is shown in Figure 5. We plot only BFGS-E since L-BFGS-E performs similarly.

If the noise is severely underestimated, it can lead to early stagnation of the algorithm due to corruption of the BFGS matrix. If the noise is severely overestimated, then the collection of non-local curvature information can result in slower progress towards the solution. Overall, the method tolerates overestimation better than underestimation of the noise level, as one would expect.

5.2. Experiments with Intermittent Noise in the Gradient. In some applications, the noise level in the gradient evaluation may fluctuate rather than remain constant. One special case is that of intermittent noise. To simulate it, we will set $\xi_f = 0$ and let the noise level in the gradient ξ_g alternate between 0 and a fixed nonzero value every N_{noise} iterations, where $N_{\text{noise}} \in \{10, 25, 50\}$. We show representative results using the `CRAGGLVY` problem in Figure 6. The `CRAGGLVY` problem is chosen because it requires more than N_{noise} iterations to solve, whereas the `ARWHEAD` problem can be solved in under 25 iterations. The noise-tolerant methods are provided the value of ϵ_g but not N_{noise}.

BFGS suffers the most from the inclusion of intermittent noise, and is unable to recover quickly enough to make progress even when there is no noise in the gradient. In contrast, BFGS-E is able to continue to make progress immediately once noise is diminished since the BFGS matrix H_k is less corrupted by noise and therefore able to take advantage of the non-noisy gradient; see in particular the stepwise behavior on the top right plot in Figure 6. L-BFGS-E performs even better than BFGS-E, but note that standard L-BFGS is quite effective when the noise toggles every $N_{\text{noise}} = 25$ or 50 iterations. This is because, if the number N_{noise} of non-noisy iterations is larger than the memory $t = 10$, L-BFGS is able to forget all noise-contaminated curvature pairs, then it is able to recover and make progress.

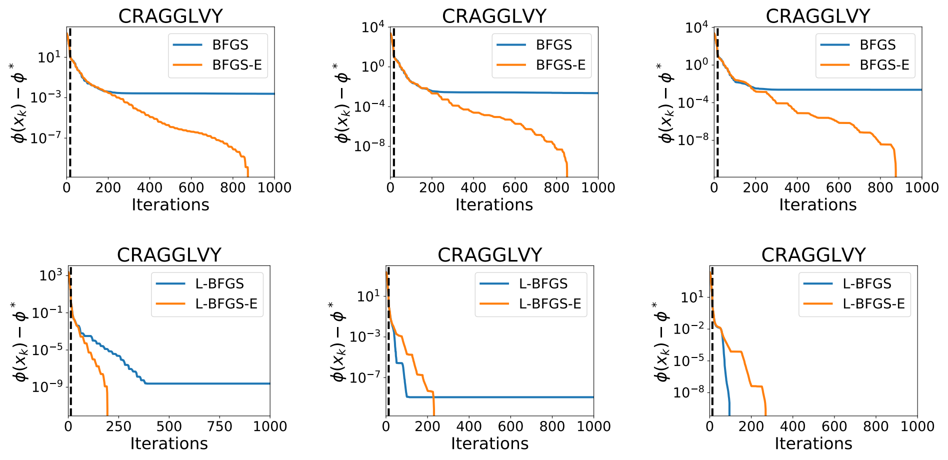

Fig. 6. *Intermittent Noise. Optimality gap $\phi(x_k) - \phi^*$ against the number of iterations on the* `CRAGGLVY` *problem. $\xi_f = 0$ and ξ_g alternates between 0 and with $\xi_g = 10^{-1}$ every N_{noise} iterations. Results for $N_{noise} = 10$ (left), 25 (middle), and 50 (right). The black dashed line denotes the iteration before the split phase becomes active.*

5.3. Comparison Against Methods that Employ Update Skipping. We now consider the performance of the BFGS (Skips) and L-BFGS (Skips) methods for constant and intermittent noise. The appeal of skipping the update when the quality of the correction pair is not assured is its economy, since the lengthening procedure involves additional gradient evaluations.

In Figures 7 and 8, we compare the performance of BFGS (Skips) and L-BFGS (Skips) to both the standard and extended methods when there is uniform constant noise in the gradient. We report the results for the `ENGVAL1` and `EIGENCLS` problems, which are of easy and medium difficulty, respectively. We chose these problems to demonstrate nuanced cases where fixing the BFGS matrix is not sufficient for making fast progress to the solution.

In Figure 7, we see that BFGS (Skips) and L-BFGS (Skips) can be much more efficient than BFGS-E and L-BFGS-E. In general, we found that methods that employ update skipping can be a strong alternative to lengthening if the problem is fairly well-conditioned and the Hessian does not change much, using much fewer gradient evaluations than the two-phase line search. However, it can fail to capture the change in curvature that is necessary for more difficult problems, such as `EIGENCLS` in Figure 8. In such cases, continuing to update the BFGS matrix using lengthening is able

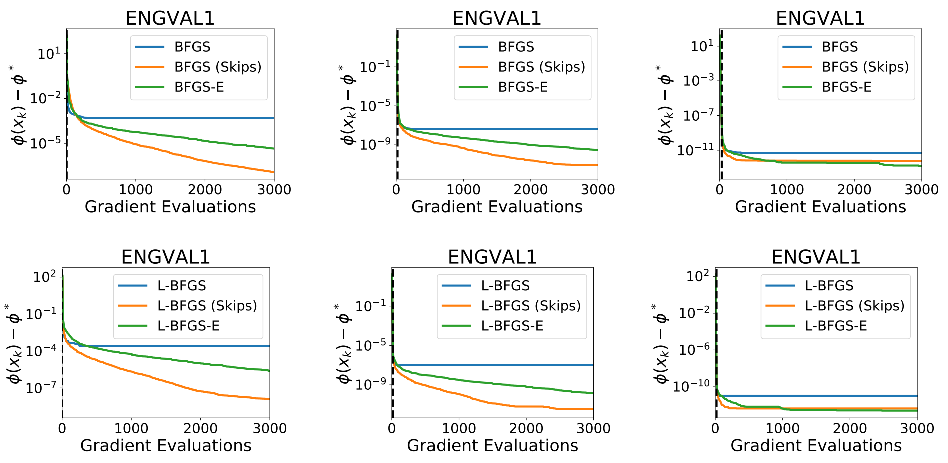

FIG. 7. *The true optimality gap $\phi(x_k) - \phi^*$ against the number of gradient evaluations on the* ***ENGVAL1*** *problem for $\epsilon_f = 0$, and for the following gradient noise levels: $\xi_g = 10^{-1}$ (left), 10^{-3} (middle), and 10^{-5} (right). The black dashed line denotes the iteration before the split phase becomes active.*

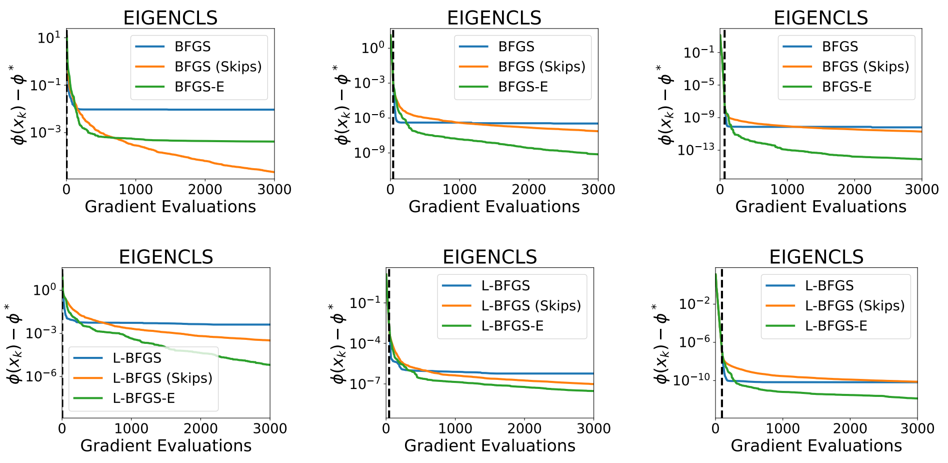

FIG. 8. *The true optimality gap $\phi(x_k) - \phi^*$ against the number of gradient evaluations on the* ***EIGENCLS*** *problem for $\epsilon_f = 0$, and for the following gradient noise levels: $\xi_g = 10^{-1}$ (left), 10^{-3} (middle), and 10^{-5} (right). The black dashed line denotes the iteration before the split phase becomes active.*

to continue to improve the quality of the Hessian approximation for more difficult problems, leading to faster decrease in the objective value compared to skipping.

To see how update skipping compares to lengthening in the intermittent setting, we report in Figure 9 results on CRAGGLVY, a problem of high difficulty. The skipping methods are able to make faster progress when noise is diminished but not as quickly as the noise-tolerant methods since they do not benefit from good updates to the BFGS matrix.

Since skipping is not as robust as lengthening for handling more difficult problems and in taking advantage of fluctuating noise, we do not report its numerical results

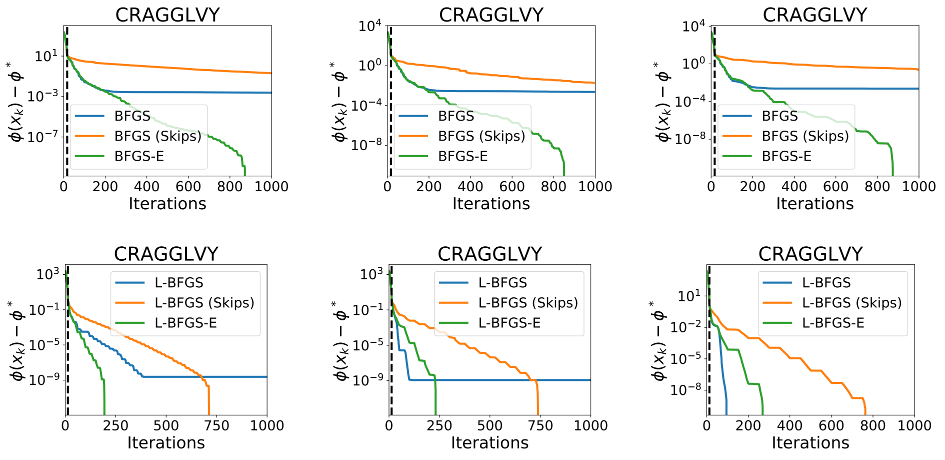

FIG. 9. *Intermittent Noise. Optimality gap $\phi(x_k) - \phi^*$ against the number of iterations on the* `CRAGGLVY` *problem. $\xi_f = 0$ and ξ_g alternates between 0 and with $\xi_g = 10^{-1}$ every N_{noise} iterations. Results for $N_{noise} = 10$ (left), 25 (middle), and 50 (right). The black dashed line denotes the iteration before the split phase becomes active.*

for the experiments in the following section.

5.4. Experiments with Function and Gradient Noise. In this set of experiments, we inject noise in both the function and gradient, i.e., $\epsilon_f, \epsilon_g > 0$. First, we report in Figures 10 and 11 results for a representative example: problem `DIXMAANH`. We ran all methods for 3000 gradient evaluations to illustrate their long term behavior, for different values of ϵ_g and ϵ_f. We note that the lengthening procedure safeguards the Hessian updating in the presence of function noise, and the relaxation in the Armijo condition (4.6) allows the methods to continue making progress far below the noise level of the function if the gradient noise is sufficiently small to provide good search directions.

Lastly, we report the performance of the methods on the 41 test problems listed in Table 2, using the profiles proposed by Morales [22]. In Figures 12 and 13 we report, respectively, the quantities

$$\log_2\left(\frac{\phi_{new} - \phi^*}{\phi_{old} - \phi^*}\right) \quad \text{and} \quad \log_2\left(\frac{evals_{new}}{evals_{old}}\right), \tag{5.2}$$

for each problem. Here ϕ_{new} and ϕ_{old} denote the true objective value of the noise-tolerant and standard methods after 3000 iterations, and $evals_{new}$ and $evals_{old}$ denote the total number of gradient evaluations required to achieve one of the conditions:

$$\phi(x_k) - \phi^* \leq \epsilon_f \quad \text{or} \quad \|\nabla\phi(x_k)\| \leq \epsilon_g. \tag{5.3}$$

All quantities are averaged over 5 runs with different seeds. In Figures 12 and 13 the problems are ordered in increasing value of the quantities given in (5.2). One can thus gauge the success of a method by the area of the graph on its side of the half-space: the larger the area, the more successful the method.

Figure 12 thus compares the (long term) ability of the methods to achieve high accuracy in the function value, whereas Figure 13 measures the short-term cost in terms of gradient evaluations to achieve the noise level in the function or gradient.

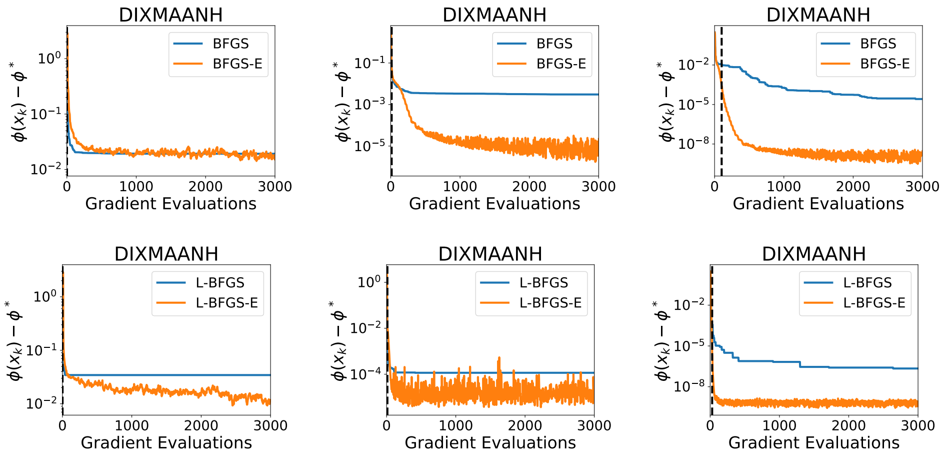

FIG. 10. *Optimality gap $\phi(x_k) - \phi^*$ against the number of gradient evaluations on problem* DIXMAANH*, with $\xi_f = 10^{-3}$ on all six plots, and with $\xi_g = 10^{-1}$ (left), $\xi_g = 10^{-3}$ (middle), and $\xi_g = 10^{-5}$ (right). The black dashed line denotes the iteration before the split phase becomes active.*

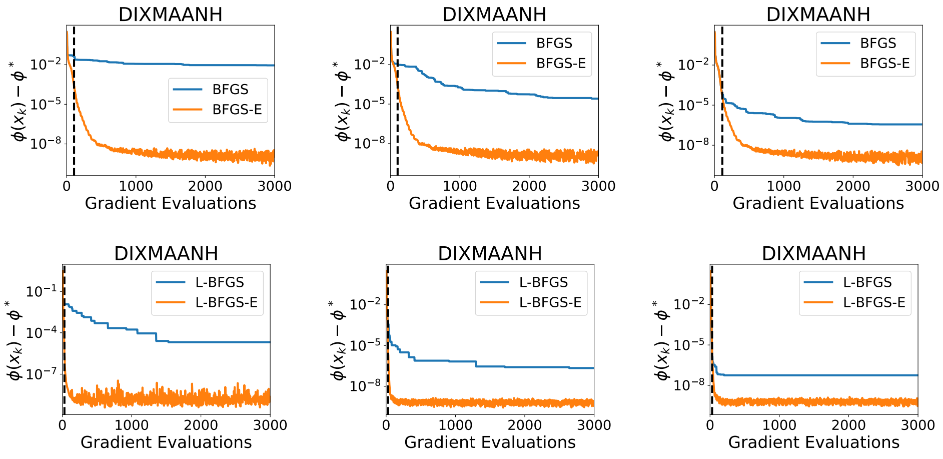

FIG. 11. *Optimality gap $\phi(x_k) - \phi^*$ against the number of gradient evaluations on problem* DIXMAANH *with $\xi_g = 10^{-5}$ on all six plots, and with $\xi_f = 10^{-1}$ (left), $\xi_f = 10^{-3}$ (middle), and $\xi_f = 10^{-5}$ (right). The black dashed line denotes the iteration before the split phase becomes active.*

These results suggest that the noise tolerant methods often provide a real improvement in the solution of certain classes of optimization problems with noisy function and gradient evaluations.

6. Final Remarks.

Although quasi-Newton methods are widely used in practice, the question of how to make BFGS and L-BFGS tolerant to errors in the function and gradient has not received sufficient attention in the literature.

This paper makes two contributions. It introduces the noise control condition (2.9), which can be used to determine when to skip a quasi-Newton update or adaptively lengthen the interval from which gradient differences can be employed reliably. Our proposed BFGS-E and L-BFGS-E methods utilize the latter and enjoy conver-

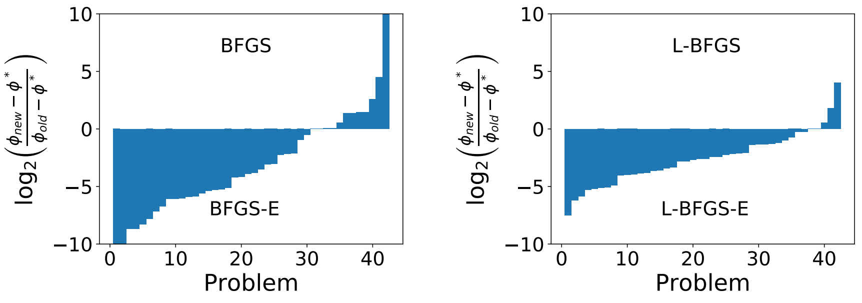

FIG. 12. *Morales profiles for the optimality gap $\phi(x_k) - \phi^*$ across 41 unconstrained CUTEst problems with $\xi_f = 10^{-3}$ and $\xi_g = 10^{-3}$. Results are averaged over 5 runs. The left figure compares BFGS against BFGS-E while the right figure compares L-BFGS against L-BFGS-E.*

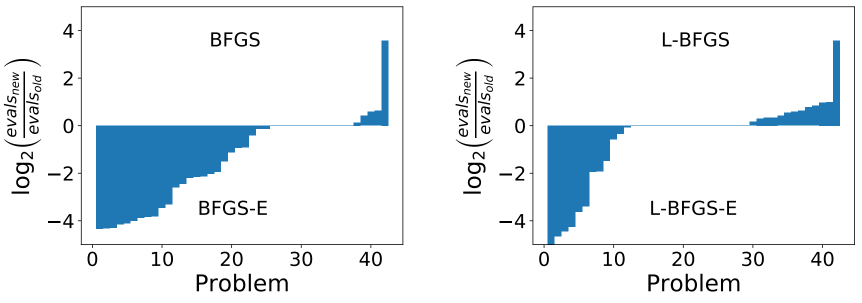

FIG. 13. *Morales profiles for the total number of gradient evaluations to achieve (5.3) across 41 unconstrained CUTEst problems with $\xi_f = 10^{-3}$ and $\xi_g = 10^{-3}$. Results are averaged over 5 runs. The left figure compares BFGS against BFGS-E while the right figure compares L-BFGS against L-BFGS-E.*

gence guarantees to a neighborhood of the solution for strongly convex functions.

The second contribution of the paper is to show that the lengthening procedure based on condition (2.9) is successful in practice, and thus transforms the theoretical algorithm proposed in [29] into a robust and practical procedure. Our numerical experiments show that quasi-Newton updating remains stable after the algorithm has reached the region where errors dominate, and this allows the noise tolerant methods to reach higher accuracy in the solution. Our testing also shows that the proposed algorithms are not more expensive than the standard BFGS and L-BFGS methods in the region where the latter two methods operate reliably. Once the iterates reach a neighborhood where BFGS updating is corrupted and the iteration stalls, the new algorithms invoke the lengthening procedure that typically requires $2 - 4$ gradient evaluations per iteration. We also tested an update skipping strategy based on the noise tolerant condition. We found that, although update skipping can be very efficient when applied to easy problems with uniform noise, the noise tolerant methods are more efficient when applied to harder problems or problems with oscillating noise.

We have made both implementations of the BFGS-E and L-BFGS-E algorithms available on GitHub[1].

[1] https://github.com/hjmshi/noise-tolerant-bfgs

Acknowledgements. We thank David Bindel, Jorge Moré, Ping Tak Peter Tang, and Andreas Waechter for their valuable input on this work and for their suggestions of problems with computational noise. We also thank Shigeng Sun and Melody Qiming Xuan for their feedback on the manuscript.

REFERENCES

[1] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *PETSc users manual*, Tech. Report Report ANL-95/11, Revision 2.1.1, Argonne National Laboratory, Argonne, Illinois, USA, 2001.

[2] R. R. Barton, *Computing forward difference derivatives in engineering optimization*, Engineering optimization, 20 (1992), pp. 205–224.

[3] A. S. Berahas, R. H. Byrd, and J. Nocedal, *Derivative-free optimization of noisy functions via quasi-newton methods*, SIAM Journal on Optimization, 29 (2019), pp. 965–993.

[4] A. S. Berahas, L. Cao, K. Choromanski, and K. Scheinberg, *Linear interpolation gives better gradients than Gaussian smoothing in derivative-free optimization*, arXiv preprint arXiv:1905.13043, (2019).

[5] A. S. Berahas, L. Cao, K. Choromanski, and K. Scheinberg, *A theoretical and empirical comparison of gradient approximations in derivative-free optimization*, arXiv preprint arXiv:1905.01332, (2019).

[6] A. S. Berahas, L. Cao, and K. Scheinberg, *Global convergence rate analysis of a generic line search algorithm with noise*, arXiv preprint arXiv:1910.04055, (2019).

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

[8] D. P. Bertsekas, *Convex Optimization Algorithms*, Athena Scientific, 2015.

[9] R. Bollapragada, D. Mudigere, J. Nocedal, H.-J. M. Shi, and P. T. P. Tang, *A progressive batching L-BFGS method for machine learning*, in International Conference on Machine Learning, 2018, pp. 620–629.

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

[11] R. H. Byrd and J. Nocedal, *A tool for the analysis of quasi-Newton methods with application to unconstrained minimization*, SIAM Journal on Numerical Analysis, 26 (1989), pp. 727–739.

[12] R. E. Caflisch, *Monte Carlo and quasi-Monte Carlo methods*, Acta numerica, 7 (1998), pp. 1–49.

[13] T. Choi and C. T. Kelley, *Superlinear convergence and implicit filtering*, SIAM Journal on Optimization, 10 (2000), pp. 1149–1162.

[14] J. Dennis and H. Walker, *Inaccuracy in quasi-Newton methods: Local improvement theorems*, in Mathematical Programming Studies, R. K. Korte B., ed., vol. 22, Springer, 1984.

[15] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, *Computing forward-difference intervals for numerical optimization*, SIAM Journal on Scientific and Statistical Computing, 4 (1983), pp. 310–321.

[16] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, London, 1981.

[17] N. I. Gould, D. Orban, and P. L. Toint, *CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization*, Computational Optimization and Applications, 60 (2015), pp. 545–557.

[18] N. I. M. Gould, D. Orban, and P. L. Toint, CUTEr *and* sifdec*: A Constrained and Unconstrained Testing Environment, revisited*, ACM Trans. Math. Softw., 29 (2003), pp. 373–394.

[19] R. M. Gower, D. Goldfarb, and P. Richtárik, *Stochastic block BFGS: squeezing more curvature out of data*, in Proceedings of the 33rd International Conference on Machine Learning, 2016.

[20] C. T. Kelley, *Implicit filtering*, vol. 23, SIAM, 2011.

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

[22] J. L. Morales and J. Nocedal, *Remark on Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization*, ACM Transactions on Mathematical Software (TOMS), 38 (2011), pp. 1–4.

[23] J. J. Moré and S. M. Wild, *Estimating computational noise*, SIAM Journal on Scientific Computing, 33 (2011), pp. 1292–1314.

[24] J. J. Moré and S. M. Wild, *Estimating derivatives of noisy simulations*, ACM Transactions

on Mathematical Software (TOMS), 38 (2012), p. 19.

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

[26] Y. Nesterov and V. Spokoiny, *Random gradient-free minimization of convex functions*, Foundations of Computational Mathematics, 17 (2017), pp. 527–566.

[27] J. Nocedal and S. Wright, *Numerical Optimization*, Springer New York, 2 ed., 1999.

[28] N. N. Schraudolph, J. Yu, and S. Günter, *A stochastic quasi-Newton method for online convex optimization*, in International Conference on Artificial Intelligence and Statistics, 2007, pp. 436–443.

[29] Y. Xie, R. H. Byrd, and J. Nocedal, *Analysis of the BFGS method with errors*, SIAM Journal on Optimization, 30 (2020), pp. 182–209.

[30] T. J. Ypma, *The effect of rounding errors on Newton-like methods*, IMA Journal of Numerical Analysis, 3 (1983), pp. 109–118.