SIAM J. OPTIM.
Vol. 29, No. 2, pp. 965–993

DERIVATIVE-FREE OPTIMIZATION OF NOISY FUNCTIONS VIA QUASI-NEWTON METHODS*

ALBERT S. BERAHAS†, RICHARD H. BYRD‡, AND JORGE NOCEDAL†

Abstract. This paper presents a finite-difference quasi-Newton method for the minimization of noisy functions. The method takes advantage of the scalability and power of BFGS updating, and employs an adaptive procedure for choosing the differencing interval h based on the noise estimation techniques of Hamming [*Introduction to Applied Numerical Analysis*, Courier Corporation, North Chelmsford, MA, 2012] and Moré and Wild [*SIAM J. Sci. Comput.*, 33 (2011), pp. 1292–1314]. This noise estimation procedure and the selection of h are inexpensive but not always accurate, and to prevent failures the algorithm incorporates a recovery mechanism that takes appropriate action in the case when the line-search procedure is unable to produce an acceptable point. A novel convergence analysis is presented that considers the effect of a noisy line-search procedure. Numerical experiments comparing the method to a function interpolating trust-region method are presented.

Key words. derivative-free optimization, nonlinear optimization, stochastic optimization

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

DOI. 10.1137/18M1177718

1. Introduction.

The BFGS method has proved to be a very successful technique for nonlinear continuous optimization, and recent work has shown that it is also very effective for nonsmooth optimization [24, 28, 29]—a class of problems for which it was not designed for and for which few would have predicted its success. Moreover, as we argue in this paper, the BFGS method with finite-difference approximations to the gradient can be the basis of a very effective algorithm for the derivative-free optimization of noisy objective functions.

It has long been recognized in some quarters [16] that one of the best methods for (nonnoisy) derivative-free optimization is the standard BFGS method with finite-difference gradients. However, its application in the noisy case has been considered problematic. The fact that finite differences can be unstable in the presence of noise has motivated the development of alternative methods, such as direct search methods [2, 13, 21, 22, 26, 30, 47] and function interpolating trust-region methods [9, 10, 12, 31, 32, 41, 42, 46], which use function evaluations at well spread out points—an indispensable feature when minimizing noisy functions. These algorithms do not, however, scale well with the number of variables.

In contrast, the L-BFGS method for deterministic optimization is able to build useful quadratic models of the objective function at a cost that is linear in the dimension n of the problem. Motivated by this observation, we propose a finite-difference quasi-Newton approach for minimizing functions that contain noise. To ensure the

*Received by the editors March 28, 2018; accepted for publication (in revised form) December 18, 2018; published electronically April 3, 2019.

http://www.siam.org/journals/siopt/29-2/M117771.html

Funding: The first author was supported by National Science Foundation grant DMS-0810213. The second author was supported by National Science Foundation grant DMS-1620070. The third author was supported by Department of Energy grant DE-FG02-87ER25047 and by DARPA grant N660011824026.

†Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60208 (albertberahas@u.northwestern.edu, j-nocedal@northwestern.edu).

‡Department of Computer Science, University of Colorado, Boulder, CO 80309 (richard@boulder.edu).

reliability of finite-difference gradients, we determine the differencing interval h based on an estimate of the noise level (i.e., the standard deviation of the noise). For this purpose, we follow the difference-table technique pioneered by Hamming [18], as improved and extended by Moré and Wild [34]. This technique samples the objective function f at a small number of equally spaced points along a random direction, and estimates the noise level from the columns of the difference table. Our optimization algorithm is adaptive, as it reestimates the noise level and differencing interval h during the course of the optimization, as necessary.

An important ingredient in the method is the line search, which in our approach serves the dual purpose of computing the length of the step (when the interval h is adequate) and determining when the differencing interval h is not appropriate and should be reestimated. When the line search is unable to find an acceptable point, the method triggers a *recovery procedure* that chooses between several courses of action. The method must, in general, be able to distinguish between the case when an unsuccessful step is due to a poor gradient approximation (in which case h may need to be increased), due to nonlinearity (in which case h should be maintained and the step length α_k shortened), or due to the confusing effects of noise.

We establish two sets of convergence results for strongly convex objective functions: one for a fixed step-length strategy and one in which the step length is computed by an Armijo backtracking line-search procedure. The latter is novel in the way it is able to account for noisy function evaluations during the line search. In both cases, we prove linear convergence to a neighborhood of the solution, where the size of the neighborhood is determined by the level of noise.

The results of our numerical experiments suggest that the proposed algorithm is competitive, in terms of function evaluations, with a well-known function interpolating trust-region method, and that it scales better with the dimension of the problem and parallelizes easily. Although the reliability of the noise estimation techniques can be guaranteed only when the noise in the objective function is independently and identically distributed, we have observed that the algorithm is often effective on problems in which this assumption is violated.

Returning to the first paragraph in this section, we note that it is the power of quasi-Newton methods, in general, rather than the specific properties of BFGS that have proven to be so effective in a surprising number of settings, including the subject of this paper. Other quasi-Newton methods could also prove to be useful. The BFGS method is appealing because it is simple, admits a straightforward extension to the large-scale setting, and is supported by a compact and elegant convergence theory.

The paper is organized into six sections. We conclude this section with some background and motivation for this work. In section 2 we compare the performance of finite-difference L-BFGS and a function interpolating trust-region method on smooth objective functions that do not contain noise. In section 3 we present the algorithm for the minimization of noisy functions, which is analyzed in section 4. In section 5 we report the results of numerical experiments, and in section 6 summarize the main findings of the paper.

1.1. Background. Although not a mainstream view, the finite-difference BFGS method is regarded by some researchers as one of the most effective methods for derivative-free optimization of smooth functions that do not contain noise, and countless users have employed it knowingly or unknowingly in that setting. To cite just one example, if a user supplies only function values, the `fminunc` MATLAB function automatically invokes a standard finite-difference BFGS method.

Nevertheless, much research has been performed in the last two decades to design other approaches for derivative-free optimization [12, 43], most prominently direct search methods [2, 13, 21, 22, 26, 30, 47] and function interpolating trust-region methods [9, 10, 12, 31, 32, 41, 42, 46]. Earlier approaches include the Nelder–Mead method [37], simulated annealing [25], and genetic algorithms [6, 20].

Function interpolating trust-region methods are more robust in the presence of noise than other techniques for derivative-free optimization. This was demonstrated by Moré and Wild [33], who report that the NEWUOA [42] implementation of the function interpolation approach was more reliable and efficient in the minimization of both smooth and noisy functions than the direct search method implemented in APPSPACK [17] and the Nelder–Mead method NMSMAX [19]. Their experiments show that direct search methods are slow and unable to scale well with the dimension of the problem; their main appeal is that they are robust due to their expansive exploration of $\mathbb{R}^n$, and are easy to analyze and implement. In spite of its efficiency, the function interpolation approach is limited by the high linear algebra cost of the iteration; straightforward implementations require $\mathcal{O}(n^6)$ operations, and although this can be reduced to $\mathcal{O}(n^4)$ operations by updating factorizations, this is still costly for high-dimensional problems [12, 42].

An established finite-difference BFGS algorithm is the *implicit filtering* method of Kelley [8, 23], which is designed for the case when noise decays as the iterates approach the solution. That method enjoys deterministic convergence guarantees to the solution, which are possible due to the assumption that noise can be diminished at any iteration, as needed. In this paper we assume that noise in the objective function does not decay to zero, and establish convergence to a neighborhood of the solution, which is the best we can hope for in this setting.

Barton [3] describes a procedure for updating the finite-difference interval h in the case when the noise is multiplicative (as is the case of round-off errors). He assumes that a bound on the noise is known, and notes that the optimal choice of h depends (in the one-dimensional case) on $|f(x)|/|f''(x)|$. Since estimating this ratio can be expensive, he updates h by observing how many digits change between $f(x_k + h)$ and $f(x_k)$, beyond the noise level. If this change is too large, h is decreased at the next iteration; if it is too small h is increased. He tested this technique successfully in conjunction with finite-difference quasi-Newton methods. In this paper, we assume that the noise level is not known and must be estimated, and discuss how to safeguard against inaccurate estimates of the noise level.

2. Optimization of smooth functions. Before embarking on our investigation of noisy objective functions we consider the case when noise is not present, and compare the performance of an L-BFGS method that uses finite differences to approximate the gradient (FDLM) and a function interpolating trust-region method, abbreviated from now on as *function interpolating* (FI) method. This will allow us to highlight the strengths of each approach and help set the stage for the investigation of the noisy case.

We write the deterministic optimization problem as

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

where $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable. In our numerical investigation, we employ the FI method of Conn, Scheinberg, and Vicente [12]. It begins by evaluating the function at $2n + 1$ points along the coordinate directions, and as the iteration progresses builds a simple quadratic model with minimum-norm Hessian. At every

iteration a new function value is computed and stored. Once $(n+1)(n+2)/2$ function values are available, a fully quadratic model $m(x)$ is constructed by interpolation. Every new iterate x_{k+1} is given by

$$x_{k+1} = \arg\min_x \{m(x) \mid \|x - x_k\|_2 \leq \Delta_k\},$$

where the trust-region radius Δ_k is updated using standard rules from derivative-free optimization [12]. We chose the method and software described in [12] because it embodies the state-of-the-art of FI methods and yet is simple enough to allow us to evaluate all its algorithmic components.

The finite-difference L-BFGS (FDLM) method is given by

$$x_{k+1} = x_k - \alpha_k H_k \nabla_h f(x_k), \tag{2.2}$$

where H_k is an approximation to the inverse Hessian, $\nabla_h f(x_k)$ is a finite-difference approximation to the gradient of f, and the step length α_k is computed by an Armijo–Wolfe line search; see [39]. We test both forward differences (FD)

$$[\nabla_{h,\text{FD}} f(x)]_i = \frac{f(x + h_{\text{FD}} e_i) - f(x)}{h_{\text{FD}}}, \quad \text{where} \quad h_{\text{FD}} = \max\{1, |x|\} (\epsilon_{\text{m}})^{1/2}, \tag{2.3}$$

and central differences (CD)

$$[\nabla_{h,\text{CD}} f(x)]_i = \frac{f(x + h_{\text{CD}} e_i) - f(x - h_{\text{CD}} e_i)}{2h_{\text{CD}}}, \quad \text{where} \quad h_{\text{CD}} = \max\{1, |x|\} (\epsilon_{\text{m}})^{1/3}. \tag{2.4}$$

Here ϵ_{m} is machine precision and $e_i \in \mathbb{R}^n$ is the ith canonical vector. (We note in passing that by employing complex arithmetic [45], highly accurate derivatives can be obtained via finite differences using a very small h. The use of complex arithmetic is, however, not always possible in practice.)

A sample of results is given in Figure 1, which plots the optimality gap $(f(x_k) - f^\star)$ versus the number of function evaluations. These four problems are from the Hock–Schittkowski collection; see [44], where their description and optimal function value $f^\star$ are given. For the FDLM method, the number of function evaluations at the kth iteration is $n + t_{\text{ls}}^k$ (FD) or $2n + t_{\text{ls}}^k$ (CD), where t_{ls}^k is the number of function evaluations performed by the Armijo–Wolfe line search at the kth iteration. The FI method is described in [12], and we refer to it as `DFOtr`; it normally computes only one new function value per iteration. As a benchmark, we also report the results of `NOMAD` [1, 2], a well-known direct-search method. It is clear that `NOMAD` is by far the slowest code.

The first problem in Figure 1 is quadratic; the FI method will generally have better performance in this case because it has finite termination while the FDLM does not. The rest of the plots in Figure 1 illustrate other behavior of the methods observed in our tests. Performance and data profiles are reported in Appendix A.1. We do not report CPU times because the FI code of Conn, Scheinberg, and Vicente [12] is not designed to be efficient in this respect, requiring $\mathcal{O}(n^6)$ flops per iteration, and is thus much slower than the FDLM method as n increases. There exist much faster FI codes (albeit much more complex), such as that of Powell [42], whose linear algebra cost is only $\mathcal{O}(n^4)$. Regardless of the implementation, scalability remains one of the main limitations of FI methods. To show that the FDLM approach can deal with problems that are out of reach for FI methods, we report in Table 1 results on the

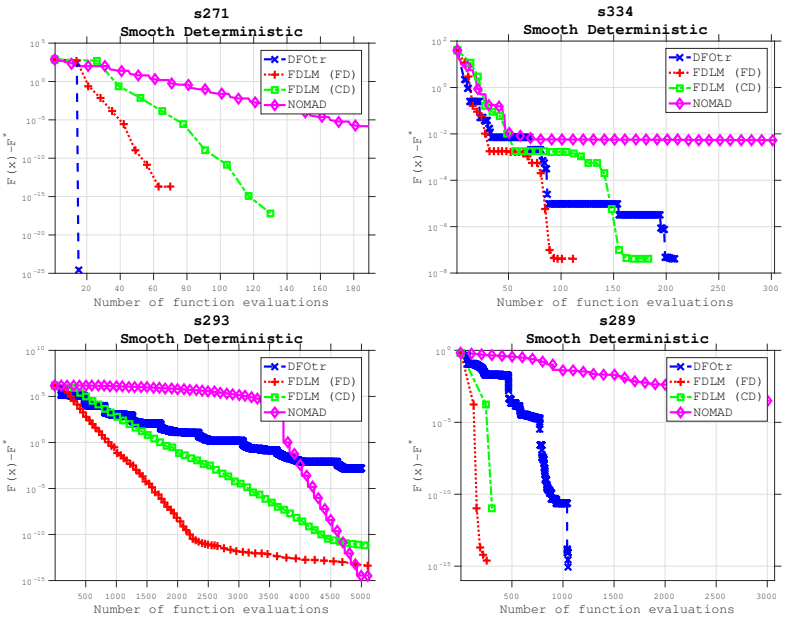

FIG. 1. *Problems without noise. Performance of the function interpolating trust-region method (*`DFOtr`*) described in* [12], `NOMAD` [1, 2]*, and the finite-difference L-BFGS method (*`FDLM`*) using forward or central differences on four problems from the Hock–Schittkowski collection* [44].

TABLE 1

CPU time (in seconds) required by the finite-difference L-BFGS method (`FDLM`*), using forward and central differences, to reach the accuracy* $f - f^\star < 10^{-6}$ *on the extended Rosenbrock function of various dimensions* n*. The results were obtained on a workstation with* 32 *GB of RAM and* 16 *Intel Xeon X*5560 *cores running at* 2.80 *GHz.*

n	10	50	100	1000	2000	5000
FD	1.8×10^{-1}	3.1×10^{-1}	7.6×10^{-1}	3.3×10^{2}	2.9×10^{3}	5.5×10^{4}
CD	2.0×10^{-1}	4.0×10^{-1}	1.2	5.6×10^{2}	5.9×10^{3}	1.0×10^{5}

extended Rosenbrock function of various dimensions (as a reference, `DFOtr` requires more than 1,800 seconds for $n = 100$).

Overall, our numerical results suggest that FDLM is a very competitive method for (nonnoisy) derivative-free optimization, particularly for large problems. Conclusive remarks about the relative performance of the FI and FDLM approaches are, however, difficult to make because there are a variety of FI methods that follow significantly different approaches than the method tested here. For example, the codes by Powell [41, 42] include two trust-region radii and employ a different procedure for placing the interpolation points. Some implementations of FI methods start with $\mathcal{O}(n^2)$ function values in order to build a quadratic model; other implementations only require $\mathcal{O}(n)$ function values to start.

2.1. Discussion. One of the appealing features of the function interpolating method is that it can move after only one function evaluation, as opposed to the $\mathcal{O}(n)$ function evaluations required per iteration by the FDLM approach. However, if the model is not accurate (or the trust region is too large) and results in an unsuccessful step, the FI approach may require many function evaluations before the

model is corrected (or the trust region is properly adjusted); this can be seen in the second, third, and fourth plots in Figure 1. On the other hand, finite-difference approximations to the gradients (2.3) and (2.4) carry some risks, as will be discussed in section 3.4. We did not encounter difficulties in our tests on smooth functions, but this is an issue that requires careful consideration in general.

The FI and FDLM methods both construct quadratic models of the objective function and compute a step as the minimizer of the model, which distinguishes them from most direct and pattern search methods. But the two methods differ significantly in nature. FI methods define the quadratic model by interpolating previously evaluated function values and (in some variants) by imposing a minimum-norm change with respect to the previous model. The estimation of the gradient and Hessian is done simultaneously, and the gradient approximation becomes accurate only when the trust region is small, typically near the solution [11]. In FI methods it is the overall quality of the model that matters. The location of the sampling points is determined by the movement of the FI algorithm. These points tend to lie along a subspace of $\mathbb{R}^n$, which can be harmful to the interpolation process. To guard against this, many FI methods include a procedure (a geometry phase) for spreading the sample points in $\mathbb{R}^n$ so that the interpolation problem is not badly conditioned.

In contrast to FI methods, the finite-difference BFGS method invests significant computation in the estimation of the gradient (n or $2n$ function evaluations) and delegates the construction of the model to BFGS updating. The function evaluations used in the estimation of the gradient parallelize easily, and the linear algebra costs are quite modest, as updating the model and computing a step can be performed in a small multiple of n using the limited memory BFGS approach [39]. Practical experience indicates that BFGS and L-BFGS give rise to well-scaled search directions that typically require little or no extra cost during the line search. Placement of the sample points is along a linearly independent set of directions (during finite differencing) and in this sense the method has some resemblance to pattern search methods, but their similarities stop there.

It is rare to find in the derivative-free optimization (DFO) literature comparisons between the finite-difference BFGS method and direct search or function interpolating trust-region methods, even when testing smooth objective functions without noise. One reason for this omission may be the perception that finite–difference-based methods are inefficient as they require at least n function evaluations per iteration, whereas other methods for DFO are more frugal in this respect. In an early paper, Powell [40] wrote in regards to derivative-free optimization, "I believe that eventually the better methods will not use derivative approximations." In this paper, we argue that a finite-difference L-BFGS method is indeed an effective technique for the minimization of certain classes of noisy functions.

These observations are a good starting point for our discussion of stochastic optimization problems.

3. Optimization of noisy functions. We study problems of the form

$$\min_{x\in\mathbb{R}^n} f(x) = \phi(x) + \epsilon(x). \tag{3.1}$$

We assume that $\phi : \mathbb{R}^n \to \mathbb{R}$ is a smooth twice continuously differentiable function and $\epsilon(x)$ is a random variable whose distribution is independent of x. (The notation $\epsilon(x)$ simply means that at any x we compute the realization of a random variable.) The model (3.1) covers the case of multiplicative noise, i.e., when $f(x) = \phi(x)(1 + \hat{\epsilon}(x))$ and $\hat{\epsilon}(x)$ is a random variable. To establish convergence results, we will assume that

$\epsilon(x)$ is independently and identically distributed and bounded, but our algorithm and presentation apply to the general model (3.1). Specifically, we are also interested in the case when the noise $\epsilon(x)$ is deterministic, as is the case of round-off errors or when adaptive algorithms are part of the function evaluation.

3.1. The finite-difference interval. Our method relies crucially on the computation of an appropriate finite-difference parameter h. It has been shown by Moré and Wild [35] that if one can estimate the level of noise in f, which we denote by σ_f, one can compute a nearly optimal h for which the error in approximating $\nabla f(x)$ is $\mathcal{O}(\sigma_f^{1/2})$ and $\mathcal{O}(\sigma_f^{2/3})$ for forward differences and central differences, respectively.

The *noise level* σ_f of the function f given in (3.1) is defined as the standard deviation of $\epsilon(x)$, i.e.,

$$\sigma_f = (\mathrm{Var}\{\epsilon(x)\})^{1/2}. \tag{3.2}$$

This quantity can be estimated using the difference table technique proposed by Hamming [18], as extended and refined by Moré and Wild [34]. We denote our *estimate* of the noise level of f by ϵ_f.

With ϵ_f in hand, we define the finite-differencing interval, as suggested in [35]. The ith component of the forward-difference approximation of the gradient of f at x is given by

$$[\nabla_{h,\mathrm{FD}} f(x)]_i = \frac{f(x + h_{\mathrm{FD}} e_i) - f(x)}{h_{\mathrm{FD}}}, \quad \text{where} \quad h_{\mathrm{FD}} = 8^{1/4} \left(\frac{\epsilon_f}{\nu_2}\right)^{1/2}, \tag{3.3}$$

and ν_2 is an estimate of $\max_{x \in [x, x+h_{\mathrm{FD}} e_i]} |\nabla^2 f(x)^T e_i|$. The central difference approximation is given by

$$[\nabla_{h,\mathrm{CD}} f(x)]_i = \frac{f(x + h_{\mathrm{CD}} e_i) - f(x - h_{\mathrm{CD}} e_i)}{2h_{\mathrm{CD}}}, \quad \text{where} \quad h_{\mathrm{CD}} = 3^{1/3} \left(\frac{\epsilon_f}{\nu_3}\right)^{1/3}, \tag{3.4}$$

and ν_3 is an estimate of the third derivative along e_i, in an interval of length $2h_{\mathrm{CD}}$ around x. Since estimating second or third derivatives along each coordinate direction is expensive, in our implementation we perform this estimation once along a random direction, as will be discussed in section 3.4. In what follows, we let $\nabla_h f(x)$ denote (3.3) or (3.4) when the distinction is not important.

3.2. Noise estimation. The noise level σ_f of a function measures the uncertainty in the computed function values, and can be estimated using Hamming's table of function differences [18]. To generate this table, Moré and Wild [34] first choose a random direction $v \in \mathbb{R}^n$ of unit norm, evaluate the function at $q+1$ equally spaced points (with spacing δ) along that ray, and compute the function differences

$$\Delta^0 f(x) = f(x), \tag{3.5a}$$

$$\Delta^{j+1} f(x) = \Delta^j[\Delta f(x)] = \Delta^j[f(x + \delta v)] - \Delta^j[f(x)], \quad j \geq 0. \tag{3.5b}$$

Let $x_i = x + u_i \delta v$, where $u_i = -q/2 + i$ for $i = 0, \ldots, q$, denote the sample points centered around x. Using the computed function differences one can build a table whose entries are

$$T_{i,j} = \Delta^j f(x_i), \qquad 1 \leq j \leq q \quad \text{and} \quad 0 \leq i \leq q - j. \tag{3.6}$$

Hamming's approach relies on the fact that differences in ϕ tend to zero rapidly, while the differences in $\epsilon(x)$ are bounded away from zero [18]. As a result, the noise level σ_f can be estimated from the mean of the squares of the columns of this difference table [34]. Specifically, for each j (where j indexes a column of the difference table and also represents the order of differencing) one defines

$$s_j^2 \stackrel{\text{def}}{=} \frac{\gamma_j}{q+1-j}\sum_{i=0}^{q-j} T_{i,j}^2, \qquad \text{where} \quad \gamma_j = \frac{(j!)^2}{(2j)!}. \tag{3.7}$$

Once an appropriate value of j is identified, the *noise estimate* ϵ_f is defined as

$$\epsilon_f \leftarrow s_j. \tag{3.8}$$

Moré and Wild [34] propose an efficient and reliable procedure, referred to as `ECnoise`, for determining the free parameters in this procedure, namely (i) the order of differencing j, (ii) the number of points q where the function is evaluated, and (iii) the spacing δ between the sample points; see section 3.4. `ECnoise` is inexpensive as it typically requires only between 4–10 function evaluations to estimate the noise level. It is available at http://www.mcs.anl.gov/~wild/cnoise/.

We should note that if the noise is stochastic one can make use of the central limit theorem and estimate the variance by sampling the function a few times (10–20) at a given point. This approach will not work, however, if the noise is deterministic and in that case we need to resort to the technique proposed by Hamming [18]. In our code we assume that the nature of the noise is not known and therefore estimate the noise using the Hamming table since it is applicable in both cases.

3.3. Specification of the finite-difference L-BFGS method. We are now ready to present the algorithm for the minimization of the function (3.1). It invokes two procedures, `LineSearch` and `Recovery`, that together produce a new iterate and, if necessary, recompute the estimate of the noise. These two procedures are described in detail after the presentation of the algorithm.

The algorithm stores the smallest function value obtained during the finite-difference gradient computation, (3.3) or (3.4). Specifically, for forward differences we define

$$f_s = \min_{x_i \in S} f(x_i), \quad \text{where} \quad S = \{x_i : x_i = x + h_{\text{FD}} e_i,\ i = 1, \ldots, n\}, \tag{3.9}$$

and let x_s denote a point where the minimum is achieved. Since S is a stencil, we refer to x_s as the *best point on the stencil*.

We now discuss the four main components of the algorithm.

3.3.1. Line search. The `LineSearch` function (line 9 of Algorithm 3.1) aims to find a step length α_k that satisfies the Armijo–Wolfe conditions,

$$f(x_k + \alpha_k d_k) \le f(x_k) + c_1 \alpha_k \nabla_h f(x_k)^T d_k \qquad \text{(Armijo condition)}, \tag{3.10a}$$

$$\nabla_h f(x_k + \alpha_k d_k)^T d_k \ge c_2 \nabla_h f(x_k)^T d_k \qquad \text{(Wolfe condition)} \tag{3.10b}$$

for some constants $0 < c_1 < c_2 < 1$. In the deterministic setting, when the gradient is exact and the objective function f is bounded below, there exist intervals of step lengths α_k that satisfy the Armijo–Wolfe conditions [39, Lemma 3.1]. However, when f is noisy, satisfying (3.10a) and (3.10b) can be problematic. To start, d_k may not

Algorithm 3.1 Adaptive finite-difference L-BFGS (FDLM).

Inputs: f (objective function), x_0 (initial iterate), $a_{\max}$ (max number of backtracks), $k \leftarrow 0$ (iteration counter), $t_{\text{count}} \leftarrow 0$ (function evaluation counter), $t_{\text{ls}} \leftarrow 0$ (function evaluation counter during `LineSearch` routine), $t_{\text{rec}} \leftarrow 0$ (function evaluation counter during `Recovery` procedure), $\zeta \in (0, 1)$ (curvature threshold).

Compute $f_0 = f(x_0)$; set $t_{\text{count}} = 1$.
Compute an estimate ϵ_f of the noise using `ECnoise` [34], at the cost of t_{ecn} function evaluations.
Update the function evaluation counter: $t_{\text{count}} = t_{\text{count}} + t_{\text{ecn}}$.
Compute h via (3.3) (FD) or (3.4) (CD).
Compute $\nabla_h f(x_0)$ using (3.3) or (3.4), and store (x_s, f_s).
$t_{\text{count}} = t_{\text{count}} + n$ (FD) or $t_{\text{count}} = t_{\text{count}} + 2n$ (CD).
while a convergence test is not satisfied **do**
    Compute $d_k = -H_k \nabla_h f(x_k)$ using the L-BFGS approach [39];
    $(x_+, f_+, \alpha_k, t_{ls}, LS_{\text{flag}}) = \texttt{LineSearch}(x_k, f_k, \nabla_h f(x_k), d_k, a_{\max})$;
    $t_{\text{count}} = t_{\text{count}} + t_{\text{ls}}$;
    if $LS_{\text{flag}} = 1$ **then** ▷ Line search failed.
        $(x_+, f_+, h, t_{\text{rec}}) = \texttt{Recovery}(f_k, x_k, d_k, h, x_s, f_s)$;
        $t_{\text{count}} = t_{\text{count}} + t_{\text{rec}}$.
    end if
    $x_{k+1} = x_+$ and $f_{k+1} = f_+$.
    Compute $\nabla_h f(x_{k+1})$ using (3.3) or (3.4), and store (x_s, f_s).
    Compute curvature pair: $s_k = x_{k+1} - x_k$ and $y_k = \nabla_h f(x_{k+1}) - \nabla_h f(x_k)$.
    Store (s_k, y_k) if $s_k^T y_k \geq \zeta \|s_k\| \|y_k\|$.
    $t_{\text{count}} = t_{\text{count}} + n$ (FD) or $t_{\text{count}} = t_{\text{count}} + 2n$ (CD).
    $k = k + 1$.
end while

be a descent direction for the smooth underlying function ϕ in (3.1), and even if it is, the noise in the objective may cause the line-search routine to make incorrect decisions. Therefore, we allow only a small number ($a_{\max}$) of line-search iterations while attempting to satisfy (3.10a) and (3.10b). We also introduce the following relaxation: if (3.10a) and (3.10b) are not satisfied at the first trial point of the line search ($\alpha_k = 1$), we relax the Armijo condition (3.10a) for subsequent trial values as follows:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + c_1 \alpha_k \nabla_h f(x_k)^T d_k + 2\epsilon_f. \tag{3.11}$$

There are three possible outcomes of the `LineSearch` function: (i) a step length α_k is found that satisfies the Armijo–Wolfe conditions, where the Armijo condition may have been relaxed as in (3.11); (ii) after $a_{\max}$ line-search iterations, the line search is able to find a step length α_k that only satisfies the relaxed Armijo condition (3.11) but not the Wolfe condition; (iii) after $a_{\max}$ line-search iterations the previous two outcomes are not achieved; we regard this as a line-search failure, set $LS_{\text{flag}} = 1$, and call the `Recovery` function to determine the cause of the failure and take corrective action.

3.3.2. Recovery mechanism. As mentioned above, the `Recovery` subroutine is the mechanism by which action is taken when the line search fails to return an acceptable point. This can occur due to the confusing effect of noisy function evaluations, a

poor gradient approximation, or high nonlinearity of the objective function (the least likely case). The procedure is described in Algorithm 3.2.

The input to the `Recovery` subroutine is the current iterate x_k and function value f_k, the search direction d_k, the current finite-difference interval h, and the best point x_s in the finite-difference stencil together with the corresponding function value f_s.

The `Recovery` routine can take three actions.

1. Return a new estimate of the differencing interval h and leave the current iterate x_k unchanged. In this case, a new noise estimate is computed along the current search direction d_k.
2. Generate a new iterate x_{k+1} without changing the differencing interval h. The new iterate is given by a small perturbation of x_k or by the best point in the stencil x_s.
3. Leave the current iterate unchanged and compute a new estimate of the noise level, along a random direction, and a new finite-difference interval h.

Algorithm 3.2 Recovery routine.

Inputs: x_k (current iterate), $f_k = f(x_k)$ (current function value), h (current finite-difference interval), $\gamma_1 \in (0,1)$, $\gamma_2 > 1$ (finite-difference interval acceptance/rejection parameters), d_k (search direction), $(x_s, f_s = f(x_s))$ (best point on the stencil), $t_{\text{rec}} \leftarrow 0$ (function evaluation counter for this routine).

Compute new noise estimate $\epsilon_f^{d_k}$ *in direction* d_k, and new $\bar{h}$ via (3.3) or (3.4).
Update function evaluation counter: $t_{\text{rec}} = t_{\text{rec}} + f_{\text{ecn}}$.
if $\bar{h} < \gamma_1 h$ OR $\bar{h} > \gamma_2 h$ **then**
    $h = \bar{h}$ and $x_+ = x_k$, $f_+ = f_k$; ▷ Case 1.
else
    $x_h = x_k + h\frac{d_k}{\|d_k\|}$; ▷ Compute small perturbation of x_k.
    compute $f_h = f(x_h)$;
    $t_{\text{rec}} = t_{\text{rec}} + 1$;
    if x_h satisfies the Armijo condition (3.10a) **then**
        $x_+ = x_h$, $f_+ = f_h$, $h = h$; ▷ Case 2.
    else
        if $f_h \leq f_s$ AND $f_h \leq f_k$ **then**
            $x_+ = x_h$, $f_+ = f_h$, $h = h$; ▷ Case 3.
        else if $f_k > f_s$ AND $f_h > f_s$ **then**
            $x_+ = x_s$, $f_+ = f_s$, $h = h$; ▷ Case 4.
        else
            $x_+ = x_k$, $f_+ = f_k$; ▷ Case 5.
            compute new noise estimate $\epsilon_f^{v_k}$ along *random direction* v_k and new $\bar{h}$;
            $h = \bar{h}$;
            $t_{\text{rec}} = t_{\text{rec}} + f_{\text{ecn}}$.
        end if
    end if
end if

Output: $x_+,\ f_+,\ h,\ t_{\text{rec}}$.

To start, the `Recovery` routine invokes the `ECnoise` procedure to compute a new noise estimate $\epsilon_f^{d_k}$ along the current search direction d_k and the corresponding differencing interval $\bar{h}$ using (3.3) or (3.4). We estimate the noise along d_k because the `Recovery` procedure may compute a new iterate along d_k, and it thus seems

natural to explore the function in that direction. If the current differencing interval h differs significantly from the new estimate $\bar{h}$, then we suspect that our current noise estimate is not reliable, and return $\bar{h}$ without changing the current iterate (Case 1). This feature is especially important when the noise is multiplicative since in this case the noise level changes over the course of optimization, and the finite-difference interval may need to be updated frequently.

On the other hand (line 5), if the two differencing intervals, h and $\bar{h}$, are similar, we regard them as reliable and assume that the line-search procedure failed due to the confusing effects of noise. We must therefore generate a new iterate by other means. We compute a small perturbation of x_k, of size h, along the current search direction d_k (line 6); if this point x_h satisfies the Armijo condition (3.10a), it is accepted, and the procedure terminates on line 10 (Case 2). Otherwise we make use of the best point x_s on the stencil. If the function value f_h at the trial point is less than both the current function values f_k and f_s, then x_h is accepted and the procedure ends on line 13 (Case 3). Else, if f_s is smaller than both f_k and f_h, we let x_s be the new iterate and terminate on line 15 (Case 4). These two cases are inspired by the global convergence properties of pattern search methods, and do not require additional function evaluations.

If none of the conditions above are satisfied, then the noise is reestimated along a random direction ($v_k \in \mathbb{R}^n$) using `ECnoise`, a new differencing interval is computed, and the `Recovery` procedure terminates without changing the iterate (Case 5). An additional action that could be taken in this last case is to switch to higher-order differences if this point of the algorithm is ever reached, as discussed in section 5.

3.3.3. Hessian approximation. Step 8 of Algorithm 3.1 computes the L-BFGS search direction. The inverse Hessian approximation H_k is updated using standard rules [39] based on the pairs $\{s_j, y_j\}$, where

$$s_k = x_{k+1} - x_k, \qquad y_k = \nabla_h f(x_{k+1}) - \nabla_h f(x_k). \tag{3.12}$$

When the line search is able to satisfy the Armijo condition but not the Wolfe condition, there is no guarantee that the product $s_k^T y_k$ is positive. In this case the pair (s_k, y_k) is discarded if the curvature condition $s_k^T y_k \geq \zeta \|s_k\| \|y_k\|$ is not satisfied for some $\zeta \in (0, 1)$.

3.3.4. Stopping tests. A variety of stopping tests have been proposed in the derivative-free optimization literature; see, e.g., [12, 21, 23, 26, 27, 42]. Here we discuss two approaches that can be employed in isolation or in combination, as no single test is best suited for all situations.

Gradient-based stopping test. One could terminate the algorithm as soon as

$$\|\nabla_h f(x_k)\|_\infty \leq \texttt{tol}, \tag{3.13}$$

where `tol` is a user-specified parameter. When using forward differences, the best one can hope for is for the norm of the gradient approximation to be $\tau(\epsilon_f)^{1/2}$, where τ depends on the norm of the second derivative. For central differences, the best accuracy is $\bar{\tau}(\epsilon_f)^{2/3}$, where $\bar{\tau}$ depends on the norm of the third derivative.

Function–value-based stopping test. One could also terminate the algorithm when

$$|f(x_k) - f(x_{k-1})| \leq \hat{\tau}\epsilon_f, \tag{3.14}$$

for $\hat{\tau} > 1$. This test is reasonable because the `ECnoise` procedure that estimates ϵ_f is scale invariant. However, there is a risk that (3.14) will trigger termination too early,

and to address this one could employ a moving average. The test can have the form

$$\frac{|f_{\mathrm{MA}}(x_k) - f(x_k)|}{|f_{\mathrm{MA}}(x_k)|} \leq \texttt{tol} \quad \text{or} \quad \frac{|f_{\mathrm{MA}}(x_k) - f(x_k)|}{\max\{1, |f_{\mathrm{MA}}(x_k)|\}} \leq \texttt{tol},$$

where $f_{\mathrm{MA}}(x_k)$ is a moving average of function values, of length M, calculated as follows. Let $f_k = f(x_k)$ and let $F^k = [f_{k-j+1}, \ldots, f_{k-1}, f_k]$ be the vector formed by the most recent function values, where $j = \min\{k+1, M\}$. We define

$$f_{\mathrm{MA}}(x_k) = \frac{1}{j} \sum_{i=1}^{j} F_i^k. \tag{3.15}$$

An alternative is to formulate the stop test as

$$|f_{\mathrm{MA}}(x_k) - f(x_k)| \leq \tau(\epsilon_f)^{1/2} \text{ for FD}, \quad \text{or} \quad |f_{\mathrm{MA}}(x_k) - f(x_k)| \leq \tau(\epsilon_f)^{2/3} \text{ for CD}.$$

3.4. Implementation of the noise estimation procedure. `ECnoise` has three parameters that if not appropriately chosen can cause `ECnoise` to fail to return a reliable estimate of the noise level: (i) the order of differencing j (see (3.8)), (ii) the number of points q used in the noise estimation, and (iii) the spacing δ between the points.

We have found the strategy proposed by Moré and Wild [34] to be effective in deciding the order of differencing j. It determines that j is appropriate if the values of σ_i surrounding σ_j are close to each other *and* if there are changes in sign among elements of the jth column of the difference table $T_{i,j}$; the latter is a clear indication that the entries of the jth column are due to noise.

Concerning the number q of sample points, we found that the values employed by `ECnoise` [34] are reliable, namely 4–8 function evaluations for stochastic noise and 6–10 for deterministic noise. However, our experience suggests that those settings may be conservative, and in our implementation use only four function evaluations for stochastic noise and six function evaluations for deterministic noise.

The finite-difference approximations (3.3) and (3.4) require a coarse estimate of the second or third derivatives (ν_2, ν_3), and are often fairly insensitive to these estimates. However, in some difficult cases poor estimates may prevent the algorithm from reaching the desired accuracy. In our implementation we estimate ν_2 using the heuristic proposed in [35, section 5, Algorithm 5.1] at the cost of 2–4 function evaluations. If this heuristic fails to provide a reasonable estimate of ν_2, we employ the finite-difference table (3.6) as a back-up to construct a rough approximation of the norm of the second derivative. In our experiments this back-up mechanism has proved to be adequate. When computing central difference approximations to the gradient, we simply set $\nu_3 \leftarrow \nu_2$, for simplicity.

4. Convergence analysis. In this section, we present two sets of convergence results for the minimization of the noisy function (3.1) under the assumption that ϕ is strongly convex. First, we analyze a method that uses a fixed step length, and then consider a more sophisticated version that employs a line search to compute the step length. The main contribution of our analysis is the inclusion of this line search and the fact that we do not assume that the errors in objective function or gradient go to zero in any deterministic or probabilistic sense; we only assume a bound on these errors. To focus on these issues, we assume that $H_k = I$, because the proof for a

general positive definite matrix H_k with bounded eigenvalues is essentially the same, but is longer and more cluttered as it involves additional constants. Convergence and complexity results for other derivative-free optimization methods can be found in [4, 15, 38].

4.1. Fixed step-length analysis. We consider the method

$$x_{k+1} = x_k - \alpha g_k, \tag{4.1}$$

where g_k stands for a finite-difference approximation to the gradient, $g_k = \nabla_h f(x_k)$, or some other approximation; our treatment is general. We define $e(x)$ to be the error in the gradient approximation, i.e.,

$$g_k = \nabla\phi(x_k) + e(x_k). \tag{4.2}$$

We should note the distinction between $e(x)$ and $\epsilon(x)$: the latter denotes the noise in the objective function (3.1).

We introduce the following assumptions to establish the first convergence result.

Assumption A.1 (strong convexity of ϕ). The function ϕ (see (3.1)) is twice continuously differentiable and there exist positive constants μ and L such that $\mu I \preceq \nabla^2\phi(x) \preceq LI$ for all $x \in \mathbb{R}^n$. (We write $\phi^\star = \phi(x^\star)$, where $x^\star$ is the minimizer of ϕ.)

Assumption A.2 (boundedness of noise in the gradient). There is a constant $\bar{\epsilon}_g > 0$ such that

$$\|e(x)\| \le \bar{\epsilon}_g \qquad \text{for all } x \in \mathbb{R}^n. \tag{4.3}$$

Assumption A.2 is satisfied if the approximate gradient is given by forward or central differences with the value of h given in (3.3) or (3.4), provided that the error $\epsilon(x)$ in the evaluation of the objection function value is bounded; see (4.30).

We now establish linear convergence to a neighborhood of the solution. Afterwards, we comment on the extension of this result to the case when a quasi-Newton iteration of the form $x_{k+1} = x_k - \alpha H_k g_k$ is used.

THEOREM 4.1. *Suppose that Assumptions* A.1 *and* A.2 *hold. Let* $\{x_k\}$ *be the iterates generated by iteration* (4.1)*, where* g_k *is given by* (4.2)*, and the step length satisfies*

$$\alpha \le 1/L. \tag{4.4}$$

Then for all k*,*

$$\phi(x_{k+1}) - \left[\phi^\star + \frac{\bar{\epsilon}_g^2}{2\mu}\right] \le (1 - \alpha\mu)\left(\phi(x_k) - \left[\phi^\star + \frac{\bar{\epsilon}_g^2}{2\mu}\right]\right), \tag{4.5}$$

where $\bar{\epsilon}_g$ *is defined in* (4.3).

Proof. Since ϕ satisfies Assumption A.1, we have, by (4.2),

$$\begin{aligned}
\phi(x_{k+1}) &\leq \phi(x_k) - \alpha \nabla\phi(x_k)^T g_k + \frac{\alpha^2 L}{2}\|g_k\|^2 \\
&= \phi(x_k) - \alpha \nabla\phi(x_k)^T \left(\nabla\phi(x_k) + e(x_k)\right) + \frac{\alpha^2 L}{2}\|\nabla\phi(x_k) + e(x_k)\|^2 \\
&= \phi(x_k) - \alpha\left(1 - \frac{\alpha L}{2}\right)\|\nabla\phi(x_k)\|^2 - \alpha(1-\alpha L)\nabla\phi(x_k)^T e(x_k) \\
&\quad + \frac{\alpha^2 L}{2}\|e(x_k)\|^2 \\
&\leq \phi(x_k) - \alpha\left(1 - \frac{\alpha L}{2}\right)\|\nabla\phi(x_k)\|^2 + \alpha(1-\alpha L)\|\nabla\phi(x_k)\|\|e(x_k)\| \\
&\quad + \frac{\alpha^2 L}{2}\|e(x_k)\|^2 \\
&\leq \phi(x_k) - \alpha\left(1 - \frac{\alpha L}{2}\right)\|\nabla\phi(x_k)\|^2 + \alpha(1-\alpha L)\left[\frac{1}{2}\|\nabla\phi(x_k)\|^2 + \frac{1}{2}\|e(x_k)\|^2\right] \\
&\quad + \frac{\alpha^2 L}{2}\|e(x_k)\|^2,
\end{aligned}$$

where the last inequality follows from the fact that $(\frac{1}{\sqrt{2}}\|\nabla\phi(x_k)\| - \frac{1}{\sqrt{2}}\|e(x_k)\|)^2 \geq 0$ and the assumption $\alpha L < 1$. Simplifying this expression, we have, for all k,

$$\phi(x_{k+1}) \leq \phi(x_k) - \frac{\alpha}{2}\|\nabla\phi(x_k)\|^2 + \frac{\alpha}{2}\|e(x_k)\|^2. \tag{4.6}$$

Since ϕ is μ-strongly convex, we can use the following relationship between the norm of the gradient squared and the distance of the kth iterate from the optimal solution:

$$\|\nabla\phi(x_k)\|^2 \geq 2\mu(\phi(x_k) - \phi^\star). \tag{4.7}$$

Together with (4.6), this yields

$$\phi(x_{k+1}) \leq \phi(x_k) - \alpha\mu(\phi(x_k) - \phi^\star) + \frac{\alpha}{2}\|e(x_k)\|^2,$$

and by (4.3),

$$\phi(x_{k+1}) - \phi^\star \leq (1 - \alpha\mu)(\phi(x_k) - \phi^\star) + \frac{\alpha}{2}\bar{\epsilon}_g^2.$$

Hence,

$$\begin{aligned}
\phi(x_{k+1}) - \phi^\star - \frac{\bar{\epsilon}_g^2}{2\mu} &\leq (1-\alpha\mu)(\phi(x_k) - \phi^\star) + \frac{\alpha}{2}\bar{\epsilon}_g^2 - \frac{\bar{\epsilon}_g^2}{2\mu} \\
&= (1-\alpha\mu)(\phi(x_k) - \phi^\star) + (\alpha\mu - 1)\frac{\bar{\epsilon}_g^2}{2\mu} \\
&= (1-\alpha\mu)\left(\phi(x_k) - \phi^\star - \frac{\bar{\epsilon}_g^2}{2\mu}\right).
\end{aligned}$$

∎

We interpret the term $[\phi^\star + \frac{\bar{\epsilon}_g^2}{2\mu}]$ in (4.5) as the best value of the objective that can be achieved in the presence of noise. Theorem 4.1 therefore establishes a Q-linear rate of convergence of $\{\phi(x_k)\}$ to that value.

Another way of stating the convergence result embodied in Theorem 4.1 is by applying the recursion to (4.5), i.e.,

$$\phi(x_k) - \left[\phi^\star + \frac{\bar{\epsilon}_g^2}{2\mu}\right] \leq (1-\alpha\mu)^k \left(\phi(x_0) - \left[\phi^\star + \frac{\bar{\epsilon}_g^2}{2\mu}\right]\right),$$

so that

$$\phi(x_k) - \phi^\star \leq (1-\alpha\mu)^k \left(\phi(x_0) - \left[\phi^\star + \frac{\bar{\epsilon}_g^2}{2\mu}\right]\right) + \frac{\bar{\epsilon}_g^2}{2\mu}. \tag{4.8}$$

This convergence result has a similar flavor to that presented in [36] for the incremental gradient method using a fixed step length; see also [7, section 4]. Note that (4.8) is an R-linear convergence result and as such is weaker than (4.5).

It is possible to prove a similar result to Theorem 4.1 for a general positive definite H_k assuming bounds on $\|H_k\|$ and $\|H_k^{-1}\|$, as would be the case with a limited memory BFGS update. However, the provable convergence rate in this case would be closer to 1, and the provable asymptotic objective value would be no smaller than the value in (4.5). This is similar to the situation in optimization without noise, where the provable convergence rate for L-BFGS is no better than for steepest descent, even though L-BFGS is superior in practice. Since this more general analysis does not provide additional insights on the main topic of this paper, we have not included it here.

4.2. Line-search analysis. In the literature of optimization of noisy functions, a number of convergence results have been established for algorithms that employ a fixed step-length strategy [5, 7, 36], but there has been little analysis of methods that use a line search.

In this section, we present a convergence result for the iteration

$$x_{k+1} = x_k - \alpha_k g_k, \tag{4.9}$$

where the step length α_k is computed by a backtracking line search governed by the relaxed Armijo condition

$$f(x_k - \alpha_k g_k) \leq f(x_k) - c_1 \alpha_k g_k^T g_k + 2\epsilon_A. \tag{4.10}$$

Here $c_1 \in (0,1)$ and $\epsilon_A > 0$ is a user-specified parameter whose choice is discussed later on. If a trial value α_k does not satisfy (4.10), the new value is set to a (fixed) fraction $\tau < 1$ of the previous value, i.e., $\alpha_k \leftarrow \tau\alpha_k$.

Our analysis relies on the following additional assumption on the error in the objective function.

Assumption A.3 (boundedness of noise in the function). There is a constant $\bar{\epsilon}_f > 0$ such that

$$|f(x) - \phi(x)| = |\epsilon(x)| \leq \bar{\epsilon}_f \qquad \text{for all } x \in \mathbb{R}^n. \tag{4.11}$$

The following result establishes linear convergence to a neighborhood of the solution.

THEOREM 4.2. *Suppose that Assumptions* A.1–A.3 *hold. Let* $\{x_k\}$ *be the iterates generated by iteration* (4.9)*, where* g_k *is given by* (4.2) *and the step length* α_k *is the maximum value in* $\{\tau^{-j};\ j = 0, 1, \dots\}$ *satisfying the relaxed Armijo condition* (4.10) *with* $\epsilon_{\mathrm{A}} > \bar{\epsilon}_f$ *and* $0 < c_1 < 1/2$*. Then, for any* $\beta \in (0, \frac{1-2c_1}{1+2c_1}]$*, we have that*

$$\phi(x_{k+1}) - [\phi^\star + \bar{\eta}] \leq \rho\left(\phi(x_k) - [\phi^\star + \bar{\eta}]\right), \quad k = 0, 1, \dots, \tag{4.12}$$

where

$$\rho = 1 - \frac{2\mu c_1 \tau (1-\beta)^2}{L}, \quad \bar{\eta} = \frac{1}{2\mu\beta^2}\bar{\epsilon}_g^2 + \frac{L}{\mu c_1 \tau (1-\beta)^2}\left(\epsilon_{\mathrm{A}} + \bar{\epsilon}_f\right), \tag{4.13}$$

and $\bar{\epsilon}_g$ *and* $\bar{\epsilon}_f$ *are defined in* (4.3) *and* (4.11)*. Additionally, if* $c_1 < 1/4$*, we can choose* $\beta = 1 - 4c_1$*, in which case,*

$$\rho = 1 - \frac{32\mu\tau c_1^3}{L}, \quad \bar{\eta} = \frac{1}{2\mu(1-4c_1)^2}\bar{\epsilon}_g^2 + \frac{L}{16\mu\tau c_1^3}\left(\epsilon_{\mathrm{A}} + \bar{\epsilon}_f\right). \tag{4.14}$$

Proof. By (4.6) in the proof of Theorem 4.1, if $\alpha \leq 1/L$, we have

$$\phi(x_k - \alpha g_k) \leq \phi(x_k) - \frac{\alpha}{2}\|\nabla\phi(x_k)\|^2 + \frac{\alpha}{2}\|e(x_k)\|^2, \tag{4.15}$$

which given Assumption A.3 implies

$$f(x_k - \alpha g_k) \leq f(x_k) - \frac{\alpha}{2}\left(\|\nabla\phi(x_k)\|^2 - \|e(x_k)\|^2\right) + 2\bar{\epsilon}_f. \tag{4.16}$$

Since we assume $\epsilon_{\mathrm{A}} > \bar{\epsilon}_f$, it is clear from comparing (4.10) and (4.16) that (4.10) will be satisfied for sufficiently small α. Thus, we have shown that the line search always finds a value of α_k such that (4.10) is satisfied.

In addition, we need to ensure that α_k is not too small when the iterates are far from $x^\star$. To this end we define

$$\beta_k = \frac{\|e(x_k)\|}{\|\nabla\phi(x_k)\|}, \tag{4.17}$$

which together with (4.2) gives

$$(1 - \beta_k)\|\nabla\phi(x_k)\| \leq \|g_k\| \leq (1 + \beta_k)\|\nabla\phi(x_k)\|. \tag{4.18}$$

Now, using (4.17) and (4.18) in (4.16) we have

$$\begin{aligned} f(x_k - \alpha g_k) &\leq f(x_k) - \frac{\alpha}{2}(1 - \beta_k^2)\|\nabla\phi(x_k)\|^2 + 2\bar{\epsilon}_f \\ &\leq f(x_k) - \frac{\alpha(1-\beta_k^2)}{2(1+\beta_k)^2}\|g_k\|^2 + 2\bar{\epsilon}_f \\ &= f(x_k) - \frac{\alpha(1-\beta_k)}{2(1+\beta_k)}\|g_k\|^2 + 2\bar{\epsilon}_f. \end{aligned}$$

Since we assume that $\epsilon_{\mathrm{A}} > \bar{\epsilon}_f$ and $c_1 < 1/2$, it is then clear that the Armijo condition (4.10) is satisfied for any $\alpha \leq 1/L$ if $(1 - \beta_k)/(1 + \beta_k) \geq 2c_1$. This is equivalent to requiring that

$$\beta_k \leq \beta \leq \frac{1 - 2c_1}{1 + 2c_1} < 1, \tag{4.19}$$

where β is an arbitrary positive constant satisfying (4.19).

Case 1. We now select such a value β and refer to iterates such that $\beta_k \leq \beta$ as Case 1 iterates. Thus, for these iterates any $\alpha \leq 1/L$ satisfies the relaxed Armijo condition (4.10), and since we find α_k using a constant backtracking factor of $\tau < 1$, we have that $\alpha_k > \tau/L$. Therefore, using Assumption A.3 and (4.18) we have

$$\begin{aligned}\phi(x_k - \alpha_k g_k) &\leq \phi(x_k) - c_1\alpha_k\|g_k\|^2 + 2\epsilon_{\mathrm{A}} + 2\bar{\epsilon}_f \\ &\leq \phi(x_k) - \frac{c_1\tau(1-\beta)^2}{L}\|\nabla\phi(x_k)\|^2 + 2\epsilon_{\mathrm{A}} + 2\bar{\epsilon}_f. \end{aligned} \tag{4.20}$$

Expression (4.20) measures the reduction in ϕ at iterates belonging to Case 1.

Case 2. We now consider iterates that do not satisfy the conditions of Case 1, namely, iterates for which $\beta_k > \beta$, or equivalently,

$$\|e(x_k)\| > \beta\|\nabla\phi(x_k)\|. \tag{4.21}$$

We have shown that the relaxed Armijo condition (4.10) is satisfied at every iteration of the algorithm. Using as before Assumption A.3, we deduce from (4.10) that

$$\begin{aligned}\phi(x_k - \alpha_k g_k) &\leq \phi(x_k) - c_1\alpha_k\|g_k\|^2 + 2\epsilon_{\mathrm{A}} + 2\bar{\epsilon}_f \\ &\leq \phi(x_k) + 2\epsilon_{\mathrm{A}} + 2\bar{\epsilon}_f. \end{aligned} \tag{4.22}$$

We now add and subtract $c_1(\tau/L)(1-\beta)^2\|\nabla\phi(x_k)\|^2$ from the right-hand side of this relation and recall (4.21) to obtain

$$\begin{aligned}\phi(x_k - \alpha_k g_k) &\leq \phi(x_k) - \frac{c_1\tau(1-\beta)^2}{L}\|\nabla\phi(x_k)\|^2 + \frac{c_1\tau(1-\beta)^2}{L}\|\nabla\phi(x_k)\|^2 + 2\epsilon_{\mathrm{A}} + 2\bar{\epsilon}_f \\ &\leq \phi(x_k) - \frac{c_1\tau(1-\beta)^2}{L}\|\nabla\phi(x_k)\|^2 + \frac{c_1\tau(1-\beta)^2}{L\beta^2}\|e(x_k)\|^2 + 2\epsilon_{\mathrm{A}} + 2\bar{\epsilon}_f \\ &\leq \phi(x_k) - \frac{c_1\tau(1-\beta)^2}{L}\|\nabla\phi(x_k)\|^2 + \frac{c_1\tau(1-\beta)^2}{L\beta^2}\bar{\epsilon}_g^2 + 2\epsilon_{\mathrm{A}} + 2\bar{\epsilon}_f \\ &= \phi(x_k) - \frac{c_1\tau(1-\beta)^2}{L}\|\nabla\phi(x_k)\|^2 + \eta, \end{aligned} \tag{4.23}$$

where

$$\eta = \frac{c_1\tau(1-\beta)^2}{L\beta^2}\bar{\epsilon}_g^2 + 2\epsilon_{\mathrm{A}} + 2\bar{\epsilon}_f. \tag{4.24}$$

Equation (4.23) establishes a recursion in ϕ for the iterates in Case 2.

Now we combine the results from the two cases: (4.20) and (4.23). Since the term multiplying $\|\nabla\phi(x_k)\|^2$ is the same in the two cases, and since $\eta \geq 2\epsilon_{\mathrm{A}} + 2\bar{\epsilon}_f$, we have that, for all k,

$$\phi(x_{k+1}) \leq \phi(x_k) - \frac{c_1\tau(1-\beta)^2}{L}\|\nabla\phi(x_k)\|^2 + \eta. \tag{4.25}$$

Subtracting $\phi^\star$ from both sides of (4.25), and using the strong convexity condition (4.7), gives

$$\phi(x_{k+1}) - \phi^\star \leq \underbrace{\left(1 - \frac{2\mu c_1\tau(1-\beta)^2}{L}\right)}_{\rho}(\phi(x_k) - \phi^\star) + \eta. \tag{4.26}$$

Clearly $0 < \rho < 1$, since the quantities $2c_1,\ \mu/L,\ \tau$, and $1-\beta$ are all less than one. We have thus shown that, for all k,

$$\phi(x_{k+1}) - \phi^\star \leq \rho(\phi(x_k) - \phi^\star) + \eta. \tag{4.27}$$

Subtracting $\eta/(1-\rho)$ from both sides, it follows that

$$\begin{aligned}\phi(x_{k+1}) - \phi^\star - \frac{\eta}{1-\rho} &\leq \rho(\phi(x_k) - \phi^\star) + \eta - \frac{\eta}{1-\rho} \\ &= \rho(\phi(x_k) - \phi^\star) - \frac{\rho\eta}{1-\rho} \\ &= \rho\left(\phi(x_k) - \phi^\star - \frac{\eta}{1-\rho}\right),\end{aligned}$$

and thus

$$\phi(x_{k+1}) - \phi^\star - \bar{\eta} \leq \rho\left(\phi(x_k) - \phi^\star - \bar{\eta}\right), \quad k = 0, 1, \ldots, \tag{4.28}$$

where $\bar{\eta} = \eta/(1-\rho)$. From (4.24), (4.26), we have that

$$\begin{aligned}\bar{\eta} &= \frac{L}{2\mu c_1 \tau(1-\beta)^2}\left(\frac{c_1\tau(1-\beta)^2}{L\beta^2}\bar{\epsilon}_g^2 + 2\epsilon_{\mathrm{A}} + 2\bar{\epsilon}_f\right) \\ &= \frac{1}{2\mu\beta^2}\bar{\epsilon}_g^2 + \frac{L}{\mu c_1\tau(1-\beta)^2}\left(\epsilon_{\mathrm{A}} + \bar{\epsilon}_f\right). \end{aligned} \tag{4.29}$$

This establishes (4.12) and (4.13).

We can obtain simpler expressions for ρ and $\bar{\eta}$ by making a particular selection of β. Recall that this parameter is required to satisfy (4.19) so that (4.28) holds for all k. In the case when $c_1 \in (0, 1/4]$, the choice $\beta = 1 - 4c_1$ will satisfy (4.19) since

$$1 - 4c_1 - \frac{1-2c_1}{1+2c_1} = \frac{1 - 4c_1 + 2c_1 - 8c_1^2 - (1-2c_1)}{1+2c_1} < 0.$$

Substituting this value of β in the definition of ρ (see (4.26)) and in (4.29) gives

$$\rho = 1 - \frac{2\mu c_1\tau(1-\beta)^2}{L} = 1 - \frac{32\mu\tau c_1^3}{L},$$

and

$$\bar{\eta} = \frac{1}{2\mu\beta^2}\bar{\epsilon}_g^2 + \frac{L}{\mu c_1\tau(1-\beta)^2}\left(\epsilon_{\mathrm{A}} + \bar{\epsilon}_f\right) = \frac{1}{2\mu(1-4c_1)^2}\bar{\epsilon}_g^2 + \frac{L}{\mu\tau 16c_1^3}\left(\epsilon_{\mathrm{A}} + \bar{\epsilon}_f\right),$$

which gives (4.14). ∎

As with (4.8) there is a different way of stating the convergence result. Applying (4.28) recursively and then moving the constant term to the right-hand side of the expression yields

$$\phi(x_k) - \phi^\star \leq \rho^k\left(\phi(x_0) - [\phi^\star + \bar{\eta}]\right) + \bar{\eta}.$$

Application of Theorem 4.2 requires a choice of the parameter β that affects the convergence constant ρ and the level of accuracy $\bar{\eta}$. One of the most intuitive expressions we could find was (4.14), which was obtained by assuming that $c_1 < 1/4$

and choosing $\beta = 1-4c_1$. This value is reasonable provided c_1 is not too small. There are other ways of choosing β when c_1 is very small; in general the theorem is stronger if β is not chosen close to zero.

In the case where $\nabla\phi(x)$ is estimated by a forward-difference approximation we can further distill our error estimate, since $\bar{\epsilon}_f$ is the source of all noise. Using Assumption A.3, it is easy to show that

$$\begin{aligned} \|e(x_k)\| = \|g_k - \nabla\phi(x_k)\| &\leq Lh/2 + 2\bar{\epsilon}_f/h \\ &= 2\sqrt{L\bar{\epsilon}_f}, \end{aligned} \tag{4.30}$$

where the equality follows by substituting $h = 2\sqrt{\bar{\epsilon}_f/L}$, which is the value that minimizes the right-hand side of the inequality. Therefore, by (4.3) we can assume that $\bar{\epsilon}_g = 2\sqrt{L\bar{\epsilon}_f}$, and the asymptotic accuracy can be estimated as

$$\begin{aligned} \bar{\eta} &= \frac{1}{2\mu\beta^2}\bar{\epsilon}_g^2 + \frac{L}{\mu c_1\tau(1-\beta)^2}\left(\epsilon_{\mathrm{A}} + \bar{\epsilon}_f\right) \qquad (4.31) \\ &= \frac{2L}{\mu\beta^2}\bar{\epsilon}_f + \frac{L}{\mu c_1\tau(1-\beta)^2}\left(\epsilon_{\mathrm{A}} + \bar{\epsilon}_f\right) \\ &= \frac{L\bar{\epsilon}_f}{\mu}\left(\frac{2}{\beta^2} + \frac{1+\theta}{c_1\tau(1-\beta)^2}\right), \qquad (4.32) \end{aligned}$$

where in the last line we have written $\epsilon_{\mathrm{A}} = \theta\bar{\epsilon}_f$ for some $\theta > 1$. If we choose $\beta = 1-4c_1$ when $c_1 < 1/4$, then

$$\bar{\eta} = \frac{L\bar{\epsilon}_f}{\mu}\left(\frac{2}{(1-4c_1)^2} + \frac{1+\theta}{16\tau c_1^3}\right). \tag{4.33}$$

Equations (4.32) and (4.33) show that the asymptotic accuracy level $\bar{\eta}$ is proportional to the bound in the error in the objective function (4.11) times the condition number. Note also that (4.31) is comparable (but larger) than the accuracy level $\bar{\epsilon}_g^2/2\mu$ in Theorem 4.1. This is not surprising as it reflects the fact that, although the line-search method can take steps that are much larger than $1/L$, the theory only uses the fact that α cannot be much smaller than $1/L$.

In the analysis presented in this section, we have chosen to analyze a backtracking line search rather than one satisfying the Armijo–Wolfe conditions (3.10a) and (3.10b), because in the strongly convex case the curvature condition (3.10b) is less critical, and because this simplifies the analysis. It is, however, possible to do a similar analysis for the Armijo–Wolfe conditions, but the algorithm and analysis would be more complex. (We should also mention that we could remove the parameter ϵ_{A} from the relaxed Armijo condition (4.10) at the cost of making the analysis more complex.)

An extension of Theorem 4.2 to a quasi-Newton iteration with positive definite H_k could be proved, but for the reasons discussed above, we have limited our analysis to the $H_k = I$ case.

5. Numerical experiments on noisy functions. In this section we present numerical results comparing the performance of the finite-difference L-BFGS method (`FDLM`, Algorithm 3.1), the function interpolating trust-region method (FI) described in [12], which we denote by `DFOtr`, and the direct search method `NOMAD` [1, 2]. We selected 49 nonlinear optimization test problems from the Hock and Schittkowski collection [44]. The distribution of problems, in terms of their dimension n, is given in Table 2.

TABLE 2
Dimensions of Hock–Schittkowski problems tested.

n	2	3	4	5	6	9	10	20	30	50	100
Number of problems	15	6	5	2	5	1	4	3	3	3	2

A limit of $100\times n$ function evaluations and 30 minutes of CPU time is given to each method. The FDLM method also terminates if one of the following two conditions holds: (i) $|f_{\text{MA}}(x_k) - f(x_k)| \leq 10^{-8}\max\{1, |f_{\text{MA}}(x_k)|\}$, where $f_{\text{MA}}(x_k)$ is defined in (3.15); or (ii) $\|\nabla_h f(x_k)\| \leq 10^{-8}$. The FI method terminates if the trust-region radius satisfies $\Delta_k \leq 10^{-8}$. The very small tolerance 10^{-8} was chosen to display the complete evolution of the runs. We ran `NOMAD` with default parameters [1].

We experimented with four different types of noise: (i) stochastic additive, (ii) stochastic multiplicative, (iii) deterministic additive, and (iv) deterministic multiplicative, and for each type of noise we considered four different noise levels. Below, we show a small sample of results for the first two types of noise.

Stochastic additive noise. The objective function has the form $f(x) = \phi(x)+\epsilon(x)$, where ϕ is a smooth function and $\epsilon(x)$ is a uniform random variable, i.e.,

$$\epsilon(x) \sim U(-\xi, \xi). \tag{5.1}$$

We investigate the behavior of the methods for noise levels corresponding to $\xi \in \{10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}\}$. In Figure 2 we report results for the four problems studied in section 2, namely `s271`, `s334`, `s293`, and `s289`, for two different noise levels (10^{-8} and 10^{-2}). The figure plots the optimality gap ($f(x_k) - \phi^\star$) versus the number of function evaluations. (For all test problems, $\phi^\star$ is known.) We note that $\phi^\star = 0$ for problems `s271`, `s293`, `s289`. The first problem, `s271` is quadratic, which is benign for `DFOtr`, which terminates as soon as a fully quadratic model is constructed.

Stochastic multiplicative noise. The objective has the form $f(x) = \phi(x)(1+\hat{\epsilon}(x))$, where ϕ is smooth and $\hat{\epsilon}(x)$ is the stochastic noise defined as in (5.1). We can write the objective in the additive form $f(x) = \phi(x) + \epsilon(x)$, where $\epsilon(x) = \phi(x)\hat{\epsilon}(x)$ varies with x, and since $\phi^\star = 0$ for problems `s271`, `s293`, and `s289`, the noise term $\epsilon(x)$ decays to zero as x_k approaches the solution; this is not the case for problem `s334`. The results are given in Figure 3.

Performance profiles. Figure 4 shows performance profiles [14] for all 49 problems, for the four different types of noise mentioned above, and for two different noise levels (10^{-8} and 10^{-2}), giving a total of 392 problems; see Appendix A.2.2 for data profiles. Since we observe from the previous results that `NOMAD` is the slowest of the methods, we do not report it in what follows. Deterministic noise was generated using the procedure proposed by Moré and Wild [33], which is described in Appendix A.2.1. For these tests, we follow [33] and use the following convergence test that measures the decrease in function value

$$f(x_0) - f(x_k) \leq (1-\tau)\left(f(x_0) - f_L\right), \tag{5.2}$$

where $\tau = 10^{-5}$, x_0 is the starting point, and f_L is the smallest value of f obtained by any solver within a given budget of $100 \times n$ function evaluations. This convergence test is well suited to derivative-free optimization because it is invariant to affine transformations and measures the function value reduction achieved relative to the best possible reduction [33].

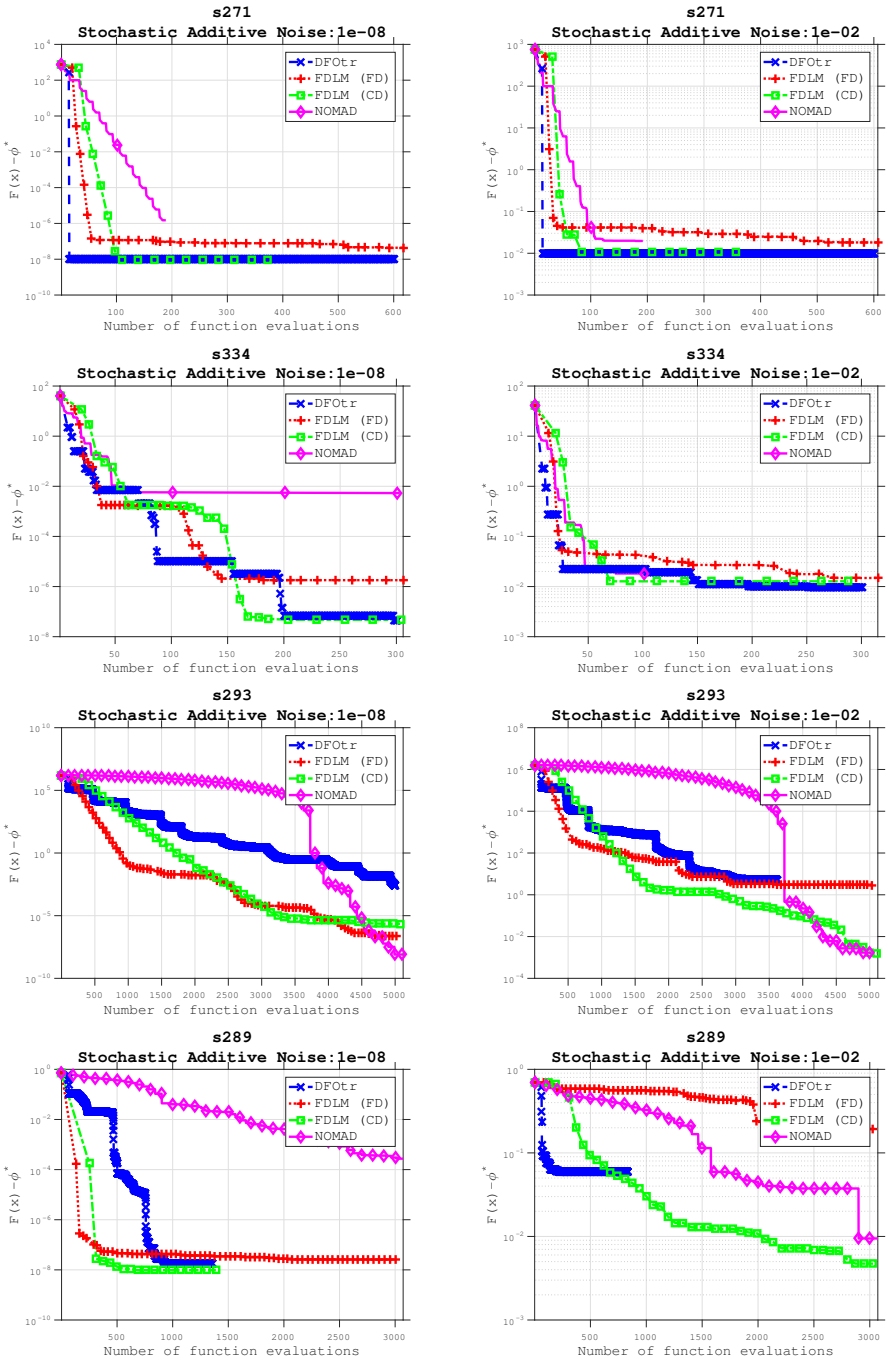

FIG. 2. Stochastic additive noise. *Performance of the function interpolating trust-region method (*`DFOtr`*) described in* [12], `NOMAD` [1, 2], *and the finite-difference L-BFGS method (*`FDLM`*) using forward or central differences. The figure plots results for four problems from the Hock–Schittkowski collection* [44], *for two different noise levels. Each row represents a different problem and each column a different noise level.*

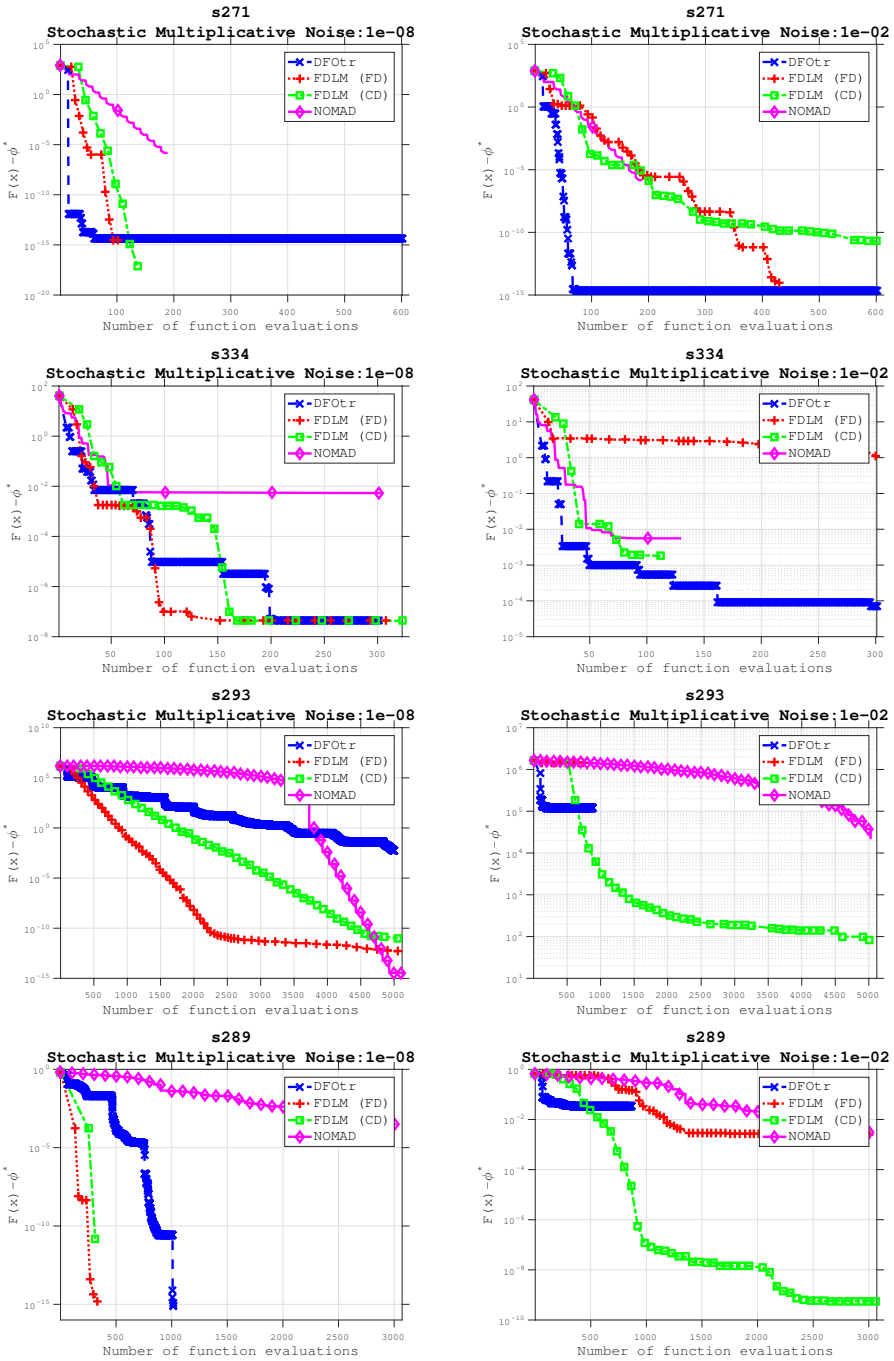

FIG. 3. Stochastic multiplicative noise. *Performance of the function interpolating trust-region method (*`DFOtr`*) described in* [12], `NOMAD` [1, 2], *and the finite-difference L-BFGS method (*`FDLM`*) using forward or central differences. The figure plots results for four problems from the Hock–Schittkowski collection* [44], *for two different noise levels. Each row represents a different problem and each column a different noise level.*

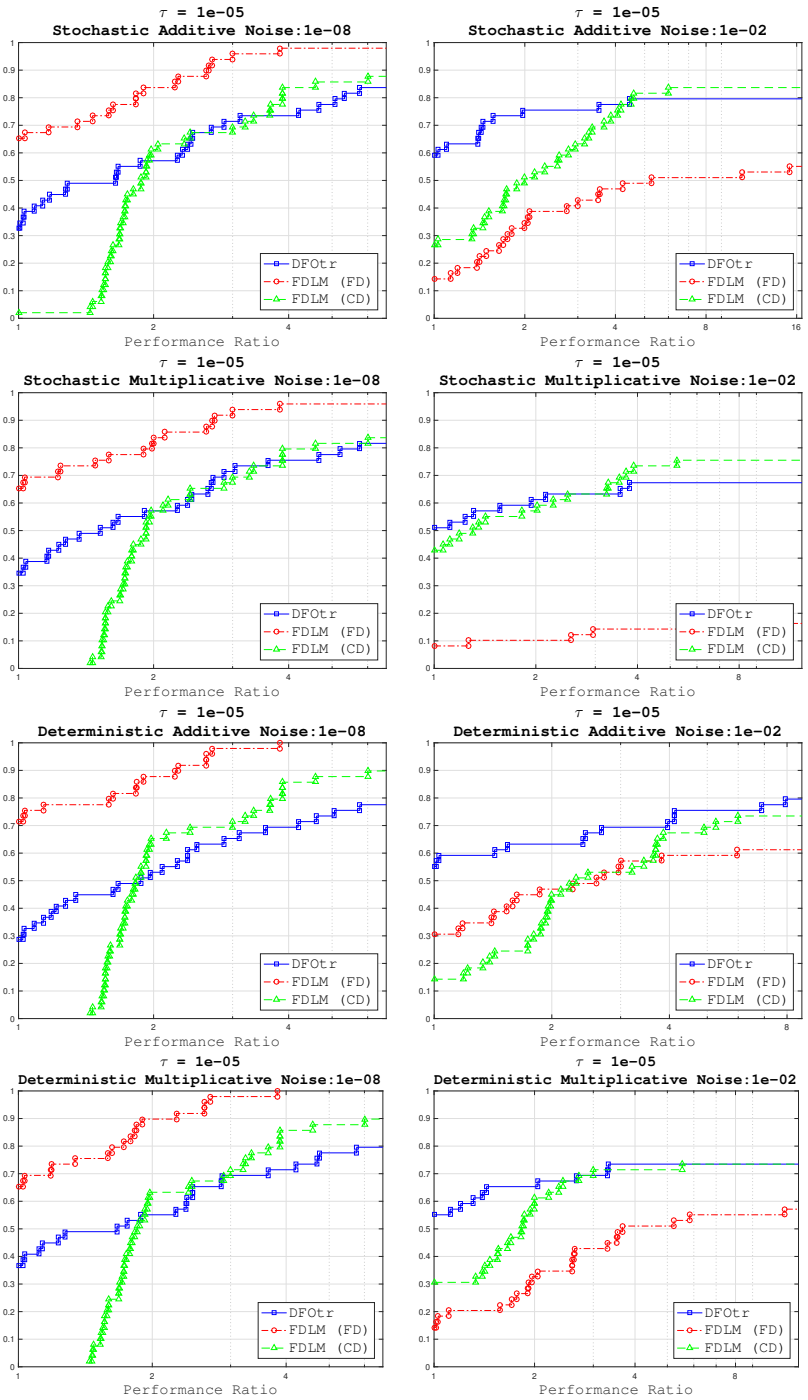

FIG. 4. *Performance profiles* ($\tau = 10^{-5}$) [14]. *Each row represents a different noise type and each column a different noise level. Column* 1: *noise level* 10^{-8}*; column* 2: *noise level* 10^{-2}. *Row* 1: *stochastic additive noise; row* 2: *stochastic multiplicative noise; row* 3: *deterministic additive noise; row* 4: *deterministic multiplicative noise.*

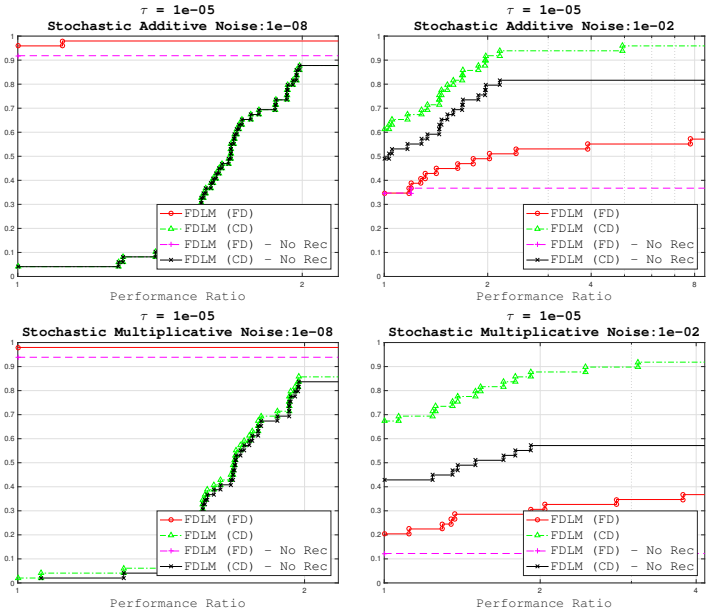

FIG. 5. *Performance of the FDLM method using forward and central differences with and without the* `Recovery` *procedure. The figure plots performance profiles for stochastic additive and stochastic multiplicative noise (noise levels* 10^{-8}, 10^{-2}*). The upper two plots show results for stochastic additive noise, and the lower two plots show results for stochastic multiplicative noise.*

Observations. Even though there is some variability, our tests indicate that overall the performances of the FI and FDLM methods are roughly comparable in terms of function evaluations. Contrary to conventional wisdom, the high per-iteration cost of finite differences is offset by the faster convergence of the FDLM method, and the potential instability of finite differences is generally not harmful in our tests. As expected, forward differences are more efficient for low levels of noise, and central differences give rise to a more robust algorithm for high noise levels (e.g., 10^{-2}); this suggests that using even higher-order gradient approximations would be useful for more difficult problems.

5.1. The recovery mechanism. We now investigate if the `Recovery` mechanism improves the robustness of our approach. We ran the FDLM method with and without the `Recovery` procedure; the latter amounts to terminating as soon as the `LineSearch` procedure fails (line 11 of Algorithm 3.1). In Figure 5 we present performance profiles for problems with stochastic additive and multiplicative noise for two noise levels (10^{-8} and 10^{-2}); see Appendix A.2.3 for data profiles. As shown, the performance of the method deteriorates substantially when the `Recovery` procedure is not used.

Our complete set of experiments shows that the `Recovery` procedure is invoked more often when the noise level is high or when forward differences are employed, as expected. It plays an important role for problems with multiplicative noise where the noise level changes during the course of the iteration and our approach relies on the `Recovery` mechanism to adjust the finite-difference interval. For some problems, the `Recovery` procedure is never invoked, but in the most difficult cases it plays a crucial role.

We also observe (see, e.g., Figures 2 and 3) that the FDLM method with forward differences is more efficient than the central difference variant in the initial stages of

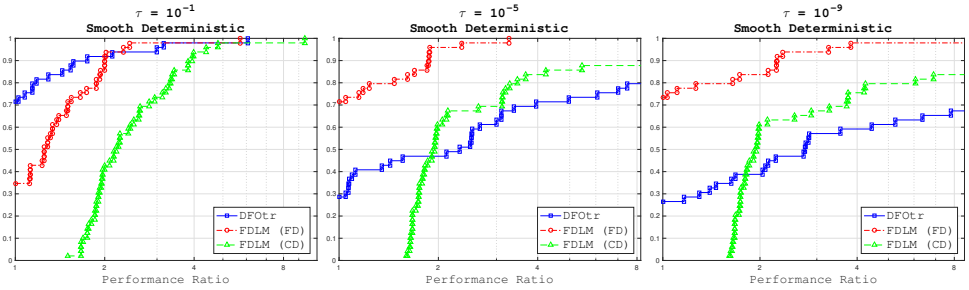

FIG. 6. *Performance profiles: smooth deterministic functions. Left:* $\tau = 10^{-1}$*; center:* $\tau = 10^{-5}$*; right:* $\tau = 10^{-9}$.

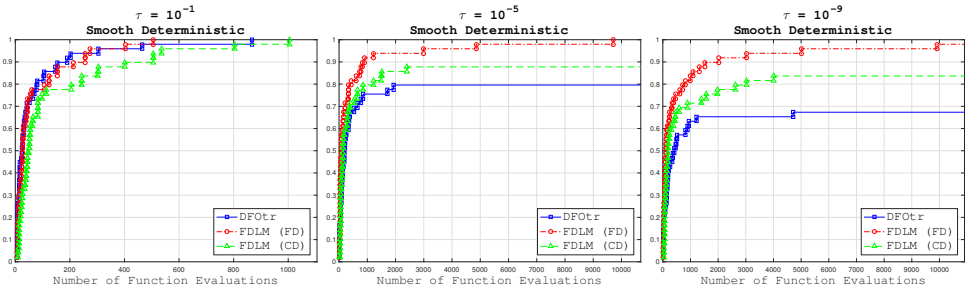

FIG. 7. *Data profiles: smooth deterministic functions. Left:* $\tau = 10^{-1}$*; center:* $\tau = 10^{-5}$*; right:* $\tau = 10^{-9}$.

the optimization, but the latter is able to achieve more accurate solutions within the given budget. It is therefore natural to consider a method that starts with forward differences and switches to central differences. How to do this in an algorithmically sound way remains a topic of investigation.

6. Final remarks. We presented a finite-difference quasi-Newton method for the optimization of noisy black-box functions. It relies on the observation that when the level of noise (i.e., its standard deviation) can be estimated, it is possible to choose finite-difference intervals that yield reliable forward or central difference approximations to the gradient. To estimate the noise level, we employ the `ECnoise` procedure of Moré and Wild [34]. Since this procedure may not always be reliable, and since the noise level may change during the course of the minimization, our algorithm includes a `Recovery` procedure that adjusts the finite-difference interval as needed. This procedure operates in conjunction with a backtracking Armijo line search.

Our numerical experiments indicate that our approach is robust and efficient. Therefore, performing $\mathcal{O}(n)$ function evaluations at every iteration to estimate the gradient is not prohibitively expensive and has the advantage that the construction of the model of the objective can be delegated to the highly scalable L-BFGS updating technique. We present a convergence analysis of our method using an Armijo-type backtracking line search that does not assume that the error in the function evaluations tends to zero.

Appendix A. Extended numerical results. In this appendix, we present additional numerical results.

A.1. Smooth functions. Figure 6 shows performance profiles for different values of τ (see (5.2)) and Figure 7 shows data profiles [33] for different values of τ.

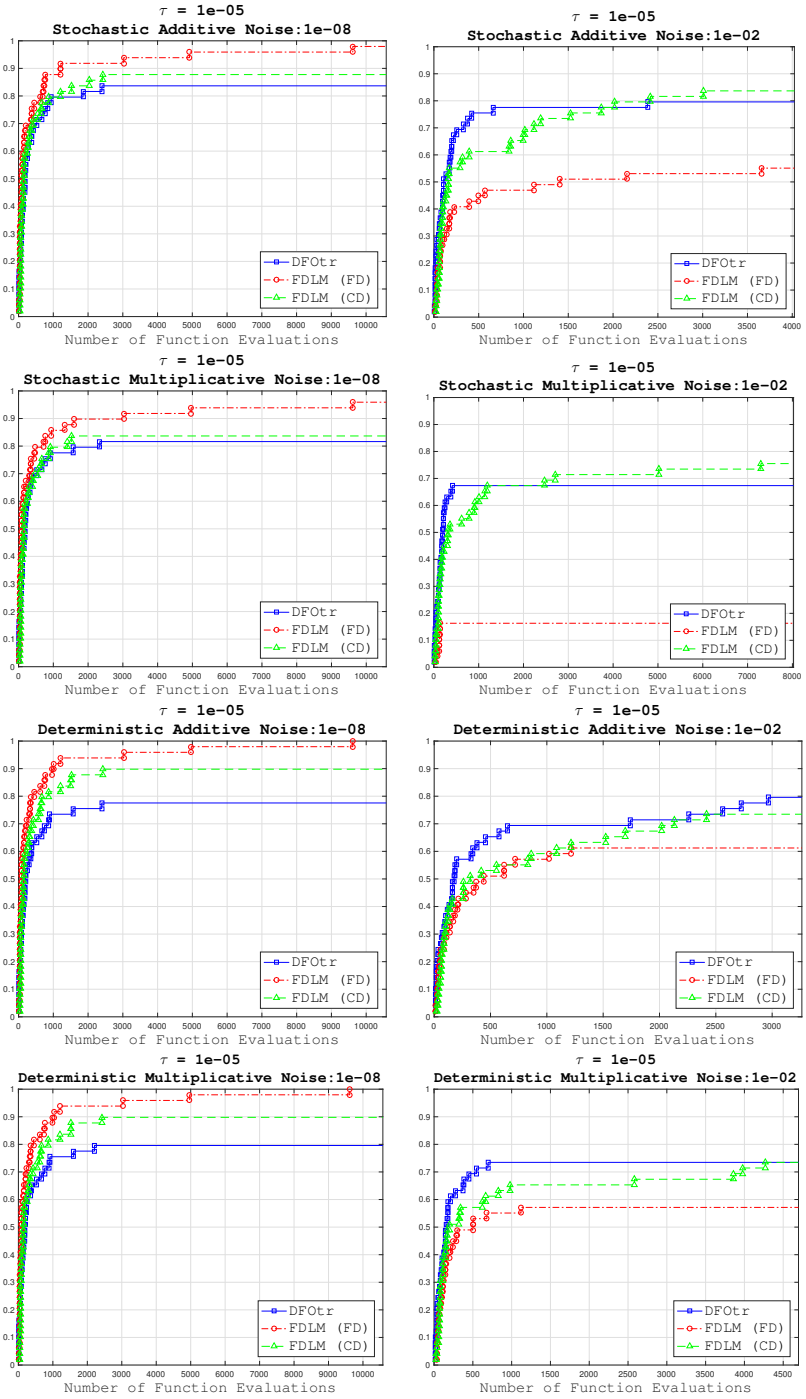

FIG. 8. *Data profiles* ($\tau = 10^{-5}$) [14]. *Each row represents a different noise type and each row column noise level. Column* 1: *noise level* 10^{-8}; *column* 2: *noise level* 10^{-2}. *Row* 1: *stochastic additive noise; row* 2: *stochastic multiplicative noise; row* 3: *deterministic additive noise; row* 4: *deterministic multiplicative noise.*

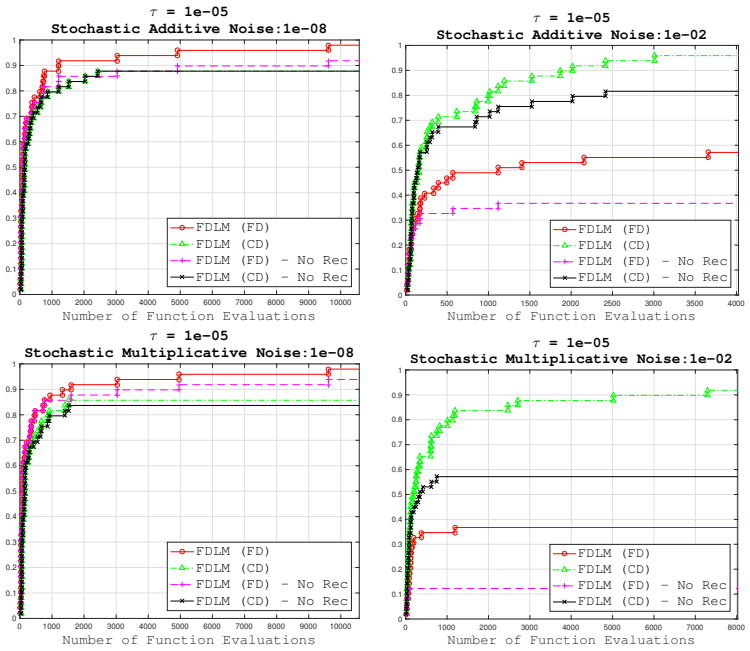

FIG. 9. *Performance of the FDLM method using forward and central differences with and without the* `Recovery` *procedure. The figure plots data profiles for stochastic additive and stochastic multiplicative noise (noise levels* 10^{-8}, 10^{-2}*). The upper two plots show results for stochastic additive noise, and the lower two plots show results for stochastic multiplicative noise.*

A.2. Noisy problems.

A.2.1. Generation of deterministic noise. Deterministic noise was generated using the procedure described by Moré and Wild [33]. Namely,

$$\epsilon(x) = \xi\psi(x),$$

where $\xi \in \{10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}\}$, and $\psi : \mathbb{R}^n \to [-1, 1]$ is defined in terms of the cubic Chebyshev polynomial $T_3(\alpha) = \alpha(4\alpha^2 - 3)$, as follows:

$$\psi(x) = T_3(\psi_0(x)), \quad \text{where} \quad \psi_0(x) = 0.9\sin(100\|x\|_1)\cos(100\|x\|_\infty) + 0.1\cos(\|x\|_2).$$

A.2.2. Data profiles. In Figure 8 we present data profiles for the problems described in section 5.

A.2.3. Performance of the recovery mechanism. We present data profiles to illustrate the performance of the FDLM method with and without the `Recovery` mechanism in Figure 9.

Acknowledgments. We are grateful to Jorge Moré and Stefan Wild for many useful suggestions and insights on noise estimation and derivative-free optimization.

REFERENCES

[1] M. A. Abramson, C. Audet, G. Couture, J. E. Dennis, Jr., S. Le Digabel, and C. Tribes, *The NOMAD project*, 2011; code available at https://www.gerad.ca/nomad.

[2] C. Audet and J. E. Dennis, Jr., *Mesh adaptive direct search algorithms for constrained optimization*, SIAM J. Optim., 17 (2006), pp. 188–217.

[3] R. R. Barton, *Computing forward difference derivatives in engineering optimization*, Eng. Optim., 20 (1992), pp. 205–224.
[4] H. H. Bauschke, W. L. Hare, and W. M. Moursi, *A derivative-free comirror algorithm for convex optimization*, Optim. Methods Softw., 30 (2015), pp. 706–726.
[5] D. P. Bertsekas, *Convex Optimization Algorithms*, Athena Scientific, Nashua, NH, 2015.
[6] A. D. Bethke, *Genetic algorithms as function optimizers*, Ph.D. Thesis, Department of Computer and Communication Sciences, University of Michigan, 1980.
[7] L. Bottou, F. E. Curtis, and J. Nocedal, *Optimization methods for large-scale machine learning*, SIAM Rev., 60 (2018), pp. 223–311.
[8] T. Choi and C. T. Kelley, *Superlinear convergence and implicit filtering*, SIAM J. Optim., 10 (2000), pp. 1149–1162.
[9] A. R. Conn, K. Scheinberg, and P. L. Toint, *On the convergence of derivative-free methods for unconstrained optimization*, in Approximation Theory and Optimization: Tributes to M. J. D. Powell, Cambridge University Press, Cambridge, 1997, pp. 83–108.
[10] A. R. Conn, K. Scheinberg, and P. L. Toint, *A derivative free optimization algorithm in practice*, in Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, American Institute of Aeronautics and Astronautics, Reston, VA, 1998, https://doi.org/10.2514/6.1998-4718.
[11] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Geometry of sample sets in derivative-free optimization: Polynomial regression and underdetermined interpolation*, IMA J. Numer. Anal., 28 (2008), pp. 721–748.
[12] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-Free Optimization*, MOS-SIAM Ser. Optim. 8, SIAM, Philadelphia, PA, 2009.
[13] J. E. Dennis, Jr., and V. Torczon, *Direct search methods on parallel machines*, SIAM J. Optim., 1 (1991), pp. 448–474.
[14] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.
[15] R. Garmanjani and L. N. Vicente, *Smoothing and worst-case complexity for direct-search methods in nonsmooth optimization*, IMA J. Numer. Anal., 33 (2013), pp. 1008–1028.
[16] N. I. M. Gould, *Private communication*, 2016.
[17] G. A. Gray and T. G. Kolda, *Algorithm* 856 : *Appspack* 4.0 : *Asynchronous parallel pattern search for derivative-free optimization*, ACM Trans. Math. Softw., 32 (2006), pp. 485–507.
[18] R. W. Hamming, *Introduction to Applied Numerical Analysis*, Courier Corporation, North Chelmsford, MA, 2012.
[19] N. J. Higham, *The Matrix Computation Toolbox*, http://www.ma.man.ac.uk/~higham/mctoolbox, 2002.
[20] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, University of Michigan Press, Ann Arbor, MI, 1975.
[21] R. Hooke and T. A. Jeeves, *"Direct search" solution of numerical and statistical problems*, J. ACM, 8 (1961), pp. 212–229.
[22] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, *Lipschitzian optimization without the Lipschitz constant*, J. Optim. Theory Appl., 79 (1993), pp. 157–181.
[23] C. T. Kelley, *Implicit Filtering*, Softw. Environ. Tools 23, SIAM, Philadelphia, PA, 2011.
[24] N. S. Keskar and A. Wächter, *A limited-memory quasi-Newton algorithm for bound-constrained non-smooth optimization*, Optim. Methods Softw., 34 (2019), pp. 150–171.
[25] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, et al., *Optimization by simulated annealing*, Science, 220 (1983), pp. 671–680.
[26] T. G. Kolda, R. M. Lewis, and V. Torczon, *Optimization by Direct Search: New perspectives on some classical and modern methods*, SIAM Rev., 45 (2003), pp. 385–482.
[27] J. Larson and S. M. Wild, *Non-intrusive termination of noisy optimization*, Optim. Methods Softw., 28 (2013), pp. 993–1011.
[28] A. S. Lewis and M. L. Overton, *Nonsmooth Optimization via BFGS*, preprint, http://www.optimization-online.org/DB_HTML/2008/12/2172.html, 2008.
[29] A. S. Lewis and M. L. Overton, *Nonsmooth optimization via quasi-newton methods*, Math. Program., 141 (2013), pp. 135–163.
[30] R. M. Lewis, V. Torczon, and M. W. Trosset, *Direct search methods: Then and now*, J. Comput. Appl. Math., 124 (2000), pp. 191–207.
[31] A. Maggiar, A. Wächter, I. S. Dolinskaya, and J. Staum, *A Derivative-Free Trust-Region Algorithm for the Optimization of Functions Smoothed via Gaussian Convolution Using Adaptive Multiple Importance Sampling*, Technical report, IEMS Department, Northwestern University, 2015.

[32] M. Marazzi and J. Nocedal, *Wedge trust region methods for derivative free optimization*, Math. Program., 91 (2002), pp. 289–305.
[33] J. J. Moré and S. M. Wild, *Benchmarking derivative-free optimization algorithms*, SIAM J. Optim., 20 (2009), pp. 172–191.
[34] J. J. Moré and S. M. Wild, *Estimating computational noise*, SIAM J. Sci. Comput., 33 (2011), pp. 1292–1314.
[35] J. J. Moré and S. M. Wild, *Estimating derivatives of noisy simulations*, ACM Trans. Math. Softw., 38 (2012), Article no. 19.
[36] A. Nedić and D. Bertsekas, *Convergence rate of incremental subgradient algorithms*, in Stochastic Optimization: Algorithms and Applications, Springer, New York, 2001, pp. 223–264.
[37] J. A. Nelder and R. Mead, *A simplex method for function minimization*, Comput. J., 7 (1965), pp. 308–313.
[38] Y. Nesterov and V. Spokoiny, *Random gradient-free minimization of convex functions*, Found. Comput. Math., 17 (2017), pp. 527–566.
[39] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed., Springer, New York, 1999.
[40] M. J. Powell, *Unconstrained Minimization Algorithms Without Computation of Derivatives*, A. E. A. Harwell Report HL72/1713, 1972.
[41] M. J. Powell, *UOBYQA: Unconstrained optimization by quadratic approximation*, Math. Program., 92 (2002), pp. 555–582.
[42] M. J. Powell, *The NEWUOA software for unconstrained optimization without derivatives*, in Large-Scale Nonlinear Optimization, Springer, New York, 2006, pp. 255–297.
[43] L. M. Rios and N. V. Sahinidis, *Derivative-free optimization: A review of algorithms and comparison of software implementations*, J. Global Optim., 56 (2013), pp. 1247–1293.
[44] K. Schittkowski, *More Test Examples for Nonlinear Programming Codes*, Lecture Notes in Econom. and Math. Systems 282, Springer, Berlin, 1987.
[45] W. Squire and G. Trapp, *Using complex variables to estimate derivatives of real functions*, SIAM Rev., 40 (1998), pp. 110–112.
[46] S. M. Wild, R. G. Regis, and C. A. Shoemaker, *ORBIT: Optimization by radial basis function interpolation in trust-regions*, SIAM J. Sci. Comput., 30 (2008), pp. 3197–3219.
[47] M. H. Wright, *Direct search methods: Once scorned, now respectable*, in Numerical Analysis: Proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis, Addison-Wesley, Reading, MA, 1996, pp. 191–208.