On the Numerical Performance of Derivative-Free Optimization Methods Based on Finite-Difference Approximations

Hao-Jun Michael Shi* Melody Qiming Xuan* Figen Oztoprak† Jorge Nocedal*

February 19, 2021

Abstract

The goal of this paper is to investigate an approach for derivative-free optimization that has not received sufficient attention in the literature and is yet one of the simplest to implement and parallelize. It consists of computing gradients of a smoothed approximation of the objective function (and constraints), and employing them within established codes. These gradient approximations are calculated by finite differences, with a differencing interval determined by the noise level in the functions and a bound on the second or third derivatives. It is assumed that noise level is known or can be estimated by means of difference tables or sampling. The use of finite differences has been largely dismissed in the derivative-free optimization literature as too expensive in terms of function evaluations and/or as impractical when the objective function contains noise. The test results presented in this paper suggest that such views should be re-examined and that the finite-difference approach has much to be recommended. The tests compared NEWUOA, DFO-LS and COBYLA against the finite-difference approach on three classes of problems: general unconstrained problems, nonlinear least squares, and general nonlinear programs with equality constraints.

Keywords: derivative-free optimization, noisy optimization, zeroth-order optimization, nonlinear optimization

*Department of Industrial Engineering and Management Sciences, Northwestern University. Shi was supported by the Office of Naval Research grant N00014-14-1-0313 P00003. Xuan was supported by the National Science Foundation grant DMS-1620022. Nocedal was supported by the Office of Naval Research grant N00014-14-1-0313 P00003, and by National Science Foundation grant DMS-1620022.
hjmshi@u.northwestern.edu, qxuan@u.northwestern.edu, j-nocedal@northwestern.edu

†Artelys Corporation. figen.topkaya@artelys.com

Contents

1 Introduction

The problem of minimizing a nonlinear objective function when gradient information is not available has received much attention in the last three decades; see [18, 30] and the references therein. A variety of methods have been developed for unconstrained optimization, and some of these methods have been extended to deal with constraints. The important benchmarking paper by Moré and Wild [36] showed that traditional methods, such as the Nelder-Mead simplex method [40] and a leading pattern-search method [26], are not competitive with the model-based trust-region approach pioneered by Powell [46] and developed concurrently by several other authors [18]. The advantages of Powell's approach reported in [36] were observed for both smooth and nonsmooth problems, as well as noisy and noiseless objective functions. Rios and Sahinidis [50] confirmed the findings of Moré and Wild concerning the inefficiency of traditional methods based on Nelder-Mead or direct searches; examples of direct-search methods are given in [1, 3, 32, 56]. Based on these studies, we regard the model-based approach of Powell as a leading method for derivative-free optimization (DFO).

There is, however, an alternative approach for DFO that is perhaps the simplest, but has been largely neglected in the nonlinear optimization literature. It consists of estimating derivatives using finite differences, and using them within standard nonlinear optimization algorithms. Many of the papers in the DFO literature dismiss this approach at the outset as being too expensive in terms of functions evaluations and/or as impractical in the presence of noise. As a result of this prevalent view, the vast majority of papers on DFO methods do not present comparisons with a finite-difference approach. We believe that if such comparisons had been made, particularly in the noiseless setting, research in the field would have followed a different path. Ironically, as pointed out by Berahas et al. [7], the finite-difference DFO approach is widely used by practitioners for noiseless problems, often unwittingly, as many established optimization codes invoke a finite-difference option when derivatives are not be provided. A disconnect between practice and research therefore occurred in the last two decades, and no systematic effort was made to bridge this gap. This paper builds upon Berahas et al. [7] and Nesterov and Spokoiny [41], and aims to bring the finite-difference DFO approach to the forefront of research by illustrating its performance on a variety of unconstrained and constrained problems, with and without noise. Our numerical experiments highlight its strengths and weaknesses, and identify some open research questions.

As is well known, the use of finite differences is delicate in the presence of noise in the function, and the straightforward application of rules designed to deal with roundoff errors, such as selecting the finite-difference interval on the order of the square-root of machine precision, leads to inefficiencies or outright failure. However, for various types of noise, the estimation of derivatives can be placed on a solid theoretical footing. One can view the task as the computation of derivatives of a smoothed function [41], or as the computation of an estimator that minimizes a mean squared error [38]. The early work by Gill et al. [23] discusses an adaptive approach for computing the difference interval in the presence of errors in the objective function, but there appears to have been no follow-up on the

application of these techniques for derivative-free optimization. The paper that influenced our work the most is by Moré and Wild [38], who describe how to choose the differencing interval as a function of the noise level and a bound on the second (or third) derivative, so as to obtain nearly optimal gradient estimates. The noise level, defined as the standard deviation of the noise, can be estimated by sampling (in the case of stochastic noise) or using a table of differences [37] (in the case of computational noise).

To test whether the finite-difference approach to DFO is competitive with established methods, for noisy and noiseless functions, we consider three classes of optimization problems: general unconstrained problems, nonlinear least-squares problems, and inequality-constrained nonlinear problems. For benchmarking, we selected the following well-established DFO methods: NEWUOA [46] for general unconstrained problems, DFO-LS [12] for nonlinear least squares, and COBYLA [44] for inequality constrained optimization. The finite-difference approach can be implemented in many standard codes for smooth deterministic optimization. We choose L-BFGS [43] for general unconstrained problems, LMDER [35] for nonlinear least squares, and KNITRO [11] for inequality-constrained optimization problems. We do not present comparisons for general problems involving both equality and inequality constraints because we were not able to find an established DFO code of such generality that was sufficiently robust in our experiments (COBYLA accepts only inequality constraints.)

Finite-difference-based methods for DFO enjoy two appealing features that motivate further research. They can easily exploit parallelism in the evaluation of functions during finite differencing, which could be critical in certain applications. In addition, finite-difference approximations can be incorporated into existing software for constrained and unconstrained optimization, sometimes rather easily. This obviates the need to redesign existing unconstrained DFO to handle more general problems, an effort that has taken two decades for interpolation-based trust-region methods [14–18, 45, 47, 48], and is yet incomplete. The strategy often suggested of simply applying an unconstrained DFO method to a penalty or augmented Lagrangian reformulation will not yield an effective general-purpose method, as is known from modern research in nonlinear programming.

1.1 Literature Review

The book by Conn, Scheinberg and Vicente [18] gives a thorough treatment of the interpolation-based trust-region approach for DFO. It presents foundational theoretical results as well as detailed algorithmic descriptions. Larson, Menickelly and Wild [30] present a comprehensive review of DFO methods as of 2018. Their survey covers deterministic and randomized methods, noisy and noiseless objective functions, problem structures such as least squares and empirical risk minimization, and various types of constraints. But they devote only two short paragraphs to the potential of finite differencing as a practical DFO method, and focus instead on complexity results for these methods. Audet and Hare [4] review direct-search and pattern-search methods, and describe a variety of practical applications solved with MADS, and NOMAD. Rios and Sahinidis [50] report extensive numerical experiments comparing many DFO codes. Neumaier [42] reviews methods endowed with convergence guarantees, with emphasis on global optimization. Nesterov and Spokoiny [41] propose a Gaussian smoothing approach for noisy DFO that involves finite-difference approximations,

and establish complexity bounds for smooth, nonsmooth, and stochastic objectives. Kelley et al. [13, 28] propose a finite-difference BFGS method, called implicit filtering, designed for the case when noise can be diminished at any iteration, as needed. In that approach, the finite-difference interval decreases monotonically. Kimiaei [29] propose a randomized method, called VSBBON that implements a noisy line search and employs quadratic models in subspaces determined adaptively.

Optimization papers that study the choice of the finite-difference interval h in the presence of errors (or noise) in the function include Gill et al. [23], which is inspired by earlier work by Lyness [33]. Gill et al. [23] gives examples where their approach fails, and pays careful attention to the estimation of bounds on the second (or third) derivatives, since these bounds are needed to obtain an accurate estimate of h. Moré and Wild [38] derive formulae for the optimal choice of h, and propose `ECnoise`, a practical procedure for estimating stochastic or deterministic noise [37]. They also give some attention to the estimation of the second derivative.

1.2 Contributions of this Paper

To our knowledge, this is the first systematic investigation into the empirical performance of finite-difference-based DFO methods relative to established techniques, across a range of problems, with and without noise in the functions. The main contributions of this paper can be summarized as follows. We use the acronym "FD-DFO method" for a method that employs some form of finite differences to approximate the gradient of the objective function and (possibly) the constraints.

- For *noiseless functions*, we found that the FD-DFO methods are at least as efficient if not superior to established methods, across all three categories of problems, without the use of sophisticated procedures for determining the finite-difference interval. Surprisingly, the carefully crafted NEWUOA code is not more efficient, as measured by the number of function evaluations, than a finite-difference L-BFGS method, and has higher linear algebra cost.

- For *noisy functions*, we observed that NEWUOA is more efficient and accurate than the finite-difference L-BFGS method for unconstrained optimization, but not by a wide margin. For least-squares problems, the performance of the recently developed DFO-LS code is comparable to that of the finite-difference version of the classical LMDER code. For inequality-constrained problems, a simple finite-difference version of KNITRO has comparable performance with COBYLA.

- The differencing formulas suggested by Moré and Wild perform well in our tests, unless the bound on the second (or third) derivative required in these formulas is poor. We found that existing procedures for estimating these bounds are not robust, and improvements on this seemingly small algorithmic detail may allow FD-DFO methods to close the performance gap observed in the unconstrained setting.

One striking observation from our study is that interpolation-based trust-region methods are more robust in the presence of noise than we expected. Although they are not simple

algorithms (e.g., they require a geometry phase), they have some appealing features. For example, NEWUOA and DFO-LS do not require knowledge of the noise level in the objective function or estimates of derivatives, and yet performed reliably for most levels of noise, suggesting that the internal logic of the algorithm normally reacts correctly to the noise inherent in the problem (although rare failures were observed). We are not aware of studies that comment on this robustness, except possibly for [36].

1.3 Limitations of this Work

For each problem class, we employed only one established DFO code to benchmark the efficiency of the corresponding FD-DFO method. We found it essential to work with a small number of codes to allow us to understand them well enough and ensure fairness of the tests. As more research is devoted to the development of FD-DFO methods, comparisons with other codes will be essential. As in most benchmarking studies, the standard disclaimer is in order: codes were tested with default options and overall performance may vary with other settings. As our focus was on scalable local optimization methods, we did not consider global optimization methods, such as Bayesian optimization, surrogate optimization, or evolutionary methods [20, 22, 27]. Most Bayesian and surrogate optimization methods are not scalable to problems above 20 variables unless modified as in TuRBO; see [21, 22]. Other approaches to global optimization include restarts, grid searches or surrogate models, enhancements that were not considered here; see [52] and [42].

Nonsmooth problems were not considered in this study. Based on the results by Lewis and Overton [31] and Curtis et al. [19], a finite-difference implementation of BFGS (not L-BFGS) could prove a strong competitor to current DFO methods. However, how to perform finite differencing robustly in the nonsmooth setting is still an open research question.

Perhaps the most important limitation of this study is that it considers only one model of noise: additive uniformly-distributed bounded noise. We do not know how the methods tested here behave for anisotropic noise, unbounded noise, or noise based on other distributions. We focused on just one model of noise because this already raised some important algorithmic questions that need to be resolved to improve the peformance of FD-DFO methods.

1.4 Organization of the Paper

A large number of experiments were performed in this study. We provide some of these numerical results in the Appendices A–C. In the main body of the paper, we display graphs or tables that attempt to accurately summarize the conclusions of the experiments. The paper is organized into five sections. Unconstrained problems are studied in section 2; nonlinear least-square problems in section 3, and nonlinear optimization problems with general inequality constraints in section 4. Concluding remarks and some open questions are described in section 5.

2 Unconstrained Optimization

In this section, we consider the solution of unconstrained optimization problems of the form

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

given only noisy evaluations,

$$f(x) = \phi(x) + \epsilon(x). \tag{2.2}$$

Here, $\epsilon(x)$ is a scalar that models deterministic or stochastic noise and ϕ is assumed to be a smooth function. We compare the performance of NEWUOA and L-BFGS with finite-difference gradients on 73 `CUTEst` problems [25], varying the dimension of each problem up to $n \leq 300$ whenever possible. All experiments were run in double precision.

We chose NEWUOA because, as mentioned above, it is regarded as one of the leading codes for unconstrained derivative-free optimization. Another well-known model-based trust-region method is DFOTR [6], which in our experience is often competitive with NEWUOA in terms of function evaluations, but has much higher per-iteration cost.

In summary, the methods tested in our experiments are as follows.

- `NEWUOA`: A model-based trust-region derivative-free algorithm that forms quadratic models using interpolation of function values, combined with a minimum Frobenius-norm update of the Hessian approximation; see Powell [46]. We called the code in Python 3.7 through PDFO developed by Ragonneau and Zhang [49]. We used the default settings in NEWUOA, which in particular, set the number of interpolation points to $2n + 1$.

- `L-BFGS`: A finite-difference implementation of the limited-memory BFGS algorithm [43] with a bisection Armijo-Wolfe line search described below. We use a memory of size $t = 10$. Forward- and central-difference options are tested. At intermediate trial points generated during the line search, the directional derivative is computed via forward- or central-differences along the direction of interest.

We first consider the case when noise is not present and later study the effect of noise on the performance of the algorithms.

2.1 Experiments on Noiseless Functions

In the first set of experiments, we let $\epsilon(x) \equiv 0$, so that the only errors in the function evaluations are due to machine precision, which we denote by ϵ_M. The approximate gradient $g(x) \in \mathbb{R}^n$ of the objective function $\phi(x)$, computed by finite differencing, is given as:

$$[g(x)]_i = \frac{f(x + h_i e_i) - f(x)}{h_i}, \qquad h_i = \max\{1, |[x]_i|\}\sqrt{\epsilon_M}; \quad \text{(forward differencing)} \tag{2.3}$$

$$[g(x)]_i = \frac{f(x + h_i e_i) - f(x - h_i e_i)}{2h_i}, \quad h_i = \max\{1, |[x]_i|\}\sqrt[3]{\epsilon_M}. \quad \text{(central differencing)} \tag{2.4}$$

Approximating the gradient through (2.3), (2.4) therefore requires $n+1$ and $2n$ function evaluations, respectively. The $\max\{1, |[x]_i|\}$ term is incorporated to handle the rounding error in $[x]_i$.

The directional derivative $D_p\phi(x) = \nabla\phi(x)^T p$ of the objective function ϕ along a direction $p \in \mathbb{R}^n$ is required within the Armijo-Wolfe line search employed by L-BFGS. It is approximated as:

$$g(x;p) = \frac{f(x+hp_u) - f(x)}{h}\|p\|, \qquad h = \sqrt{\epsilon_M}; \qquad \text{(forward differencing)} \tag{2.5}$$

$$g(x;p) = \frac{f(x+hp_u) - f(x-hp_u)}{2h}\|p\|, \qquad h = \sqrt[3]{\epsilon_M}; \qquad \text{(central differencing)} \tag{2.6}$$

where $p_u = p/\|p\|$ is the normalized direction.

These choices of h can be improved by including contributions of the second and third derivatives, respectively, as discussed in the next subsection. However, we find that in the noiseless setting, and for our test functions, such a refinement is not needed to make the finite-difference L-BFGS approach competitive.

The algorithms are terminated when either

$$\phi(x_k) - \phi^* \leq \tau \cdot \max\{1, |\phi^*|\}, \tag{2.7}$$

with $\tau = 10^{-6}$, or when the limit of $500 \times n$ function evaluations is reached. The optimal value ϕ^* is determined by running BFGS with *exact* gradients (provided by `CUTEst` [25]) until no more progress can be made on the function. Complete numerical results are given in Tables 4-8 in Appendix C.1.1.

In Figure 2.1, we summarize the results using *log-ratio profiles* proposed by Morales [34], which in this case report the quantity

$$\log_2\left(\frac{\text{evals}_{\text{LBFGS}}}{\text{evals}_{\text{NEWUOA}}}\right), \tag{2.8}$$

where $\text{evals}_{\text{LBFGS}}$ and $\text{evals}_{\text{NEWUOA}}$ denote the total number of function evaluations for NEWUOA and L-BFGS to satisfy (2.7) or reach the maximum number of function evaluations. In the figures, the ratios (2.8) are plotted in increasing order. Thus, the area of the shaded region gives an idea of the general success of a method.

For fair comparison, the runs with the following outcomes were not included in Figure 2.1. There are six instances where NEWUOA's initial sampling of $2n+1$ points immediately discovers a point whose function value satisfies (2.7), thirteen instances where L-BFGS and NEWUOA converged to different minimizers, and two instances where either one of the methods failed.

Overall, we observe in these tests that forward-difference L-BFGS outperforms NEWUOA on the majority of problems in terms of function evaluations. This is perhaps surprising because NEWUOA was designed to be parsimonious in terms of function evaluations, whereas finite-difference L-BFGS requires n function evaluations per iteration. It is also notable that this is achieved without any additional information (for example, the Lipschitz constant of the gradient for forward differencing) to squeeze out the best possible accuracy of the

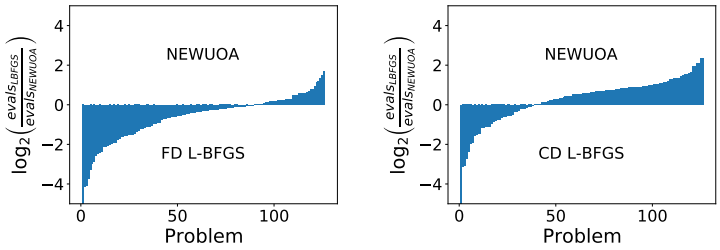

Figure 1: *Efficiency, Noiseless Case.* Log-ratio profiles for the total number of function evaluations to achieve (2.7) with $\epsilon(x) = 0$. The left figure compares forward-difference L-BFGS with NEWUOA, and the right figure compares central-difference L-BFGS with NEWUOA.

gradient in the finite-difference L-BFGS approach. As expected, central-difference L-BFGS requires significantly more function evaluations than the forward-difference option, and does not provide significant benefit in terms of solution accuracy for the majority of the problems. In particular, when run in double precision, forward-difference L-BFGS is able to converge to the same tolerance that one would expect with analytical gradients.

In order to illustrate iteration cost, we report in Table 1 the CPU times for NEWUOA and L-BFGS for a representative problem as the number of variables n increases. NEWUOA's execution time grows much faster than L-BFGS because one iteration of NEWUOA requires $O(n^2)$ flops, whereas the iteration cost of L-BFGS is $4tn$ flops, where $t = 10$ and all test functions are inexpensive to evaluate. Across all the problems, we observe that when $n \approx 100$, NEWUOA can take at least $5 - 10$ times longer than L-BFGS in terms of wall-clock time.

n	10	20	30	50	90	100	500
NEWUOA	9.61e-2	9.99e-2	1.21e-1	1.99e-1	5.83e-1	8.92e-1	2.42e2
FD L-BFGS	1.13e-2	1.68e-2	2.26e-2	3.18e-2	5.50e-2	6.67e-2	4.67e-1
CD L-BFGS	1.30e-2	2.09e-2	2.90e-2	4.28e-2	6.63e-2	8.56e-2	6.63e-1

Table 1: *Computing Time, Noiseless Case.* CPU time (in seconds) required for NEWUOA and L-BFGS with forward (FD) and central (CD) differencing to satisfy (2.7) on the `NONDIA` problem, as the dimension of the problem n increases.

While NEWUOA is an inherently sequential algorithm, finite-difference L-BFGS offers ample opportunities for parallelism when computing the finite-difference approximation to the gradient, which can be distributed across multiple nodes, with the only sequential bottleneck arising in the function evaluations within Armijo-Wolfe line search. Given the ratio between the number of function evaluations needed for NEWUOA versus finite-difference L-BFGS reported in Tables 4-8 in Appendix C.1.1, we hypothesize that even small amounts of parallelism across even a few processors may lead to significant speedup of L-BFGS. This is an often overlooked benefit of finite-difference-based methods in the DFO literature, as we are not aware of implementations of model-based trust-region methods that benefit significantly from parallelism.

We conclude this subsection by noting that most DFO methods were developed and tested primarily in the noiseless setting [18, 30]. Our results suggest that most of these methods may not be competitive with the finite-difference L-BFGS approach in the noiseless case. However, since no single set of experiments can conclusively establish such a claim, tests by other researchers should verify our observations.

2.2 Experiments on Noisy Functions

In this set of experiments, we synthetically inject uniform stochastic noise into the objective function. In particular, we sample $\epsilon(x) \sim \sigma_f U(-\sqrt{3}, \sqrt{3})$ i.i.d. independent of x, where $\sigma_f \in \{10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}\}$. (Thus, strictly speaking we should write ϵ, but we keep the notation $\epsilon(x)$ for future generality.) By construction of $\epsilon(x)$, we have that $\sigma_f^2 = \mathbb{E}[\epsilon(x)^2]$. We refer to standard deviation of the noise σ_f as the *noise level*.

We employ a more precise formula for the finite-difference interval than in the noiseless setting — one that depends both on the noise level σ_f and on the curvature of the function. Specifically, we compute a different h_i for each coordinate direction based on the following well-known result [38] that provides bounds on the mean-squared error of estimated derivatives.

Theorem 2.1. *Let $h_0 > 0$ and $x, p \in \mathbb{R}^n$, with $\|p\| = 1$, be given. Define the interval $I_{FD} = \{x + tp : t \in [0, h_0]\}$. If $|D_p^2\phi(y)| = |p^T \nabla^2 \phi(y) p| \leq L$ for all $y \in I_{FD}$, then for any $\tilde{h} \in (0, h_0]$, the following bound holds for forward differencing:*

$$\sigma_{g,p}(\tilde{h})^2 \equiv \mathbb{E}\left[\Big(g(x; p, \tilde{h}) - D_p\phi(x)\Big)^2\right] \leq \frac{L^2\tilde{h}^2}{4} + \frac{2\sigma_f^2}{\tilde{h}^2}, \tag{2.9}$$

where

$$g(x; p, \tilde{h}) = \frac{f(x + \tilde{h}p) - f(x)}{\tilde{h}}. \tag{2.10}$$

Similarly for central differencing, if the third derivative satisfies $|D_p^3\phi(y)| \leq M$ for all $y \in I_{CD} = \{x \pm tp : t \in [0, h_0]\}$, then for any $\tilde{h} \in (0, h_0]$ we have

$$\sigma_{g,p}(\tilde{h})^2 \equiv \mathbb{E}[(g(x; p, \tilde{h}) - D_p\phi(x))^2] \leq \frac{M^2\tilde{h}^4}{36} + \frac{\sigma_f^2}{2\tilde{h}^2}, \tag{2.11}$$

where $D_p^3\phi(y)$ denotes the third directional derivative with respect to direction p and

$$g(x; p, \tilde{h}) = \frac{f(x + \tilde{h}p) - f(x - \tilde{h}p)}{\tilde{h}}. \tag{2.12}$$

By minimizing the upper bounds in (2.9), (2.11) with respect to $\tilde{h}$, one obtains the following expressions for each coordinate direction [38],

$$[g(x)]_i = \frac{f(x + h_i e_i) - f(x)}{h_i}, \qquad h_i = \sqrt[4]{8}\sqrt{\frac{\sigma_f}{L_i}}; \quad \text{(forward differencing)} \tag{2.13}$$

$$[g(x)]_i = \frac{f(x + h_i e_i) - f(x - h_i e_i)}{2h_i}, \quad h_i = \sqrt[3]{\frac{3\sigma_f}{M_i}}, \quad \text{(central differencing)} \tag{2.14}$$

where L_i and M_i are bounds on the second and third derivative along the i-th coordinate direction e_i. The noise level of the derivative along an arbitrary direction $p \in \mathbb{R}^n$ can be shown to be

$$\sigma_{g,p} \leq \sqrt[4]{2}\sqrt{L\sigma_f} \qquad \sigma_{g,p} \leq \frac{\sqrt[6]{3}}{2} M^{1/3}\sigma_f^{2/3},$$

for forward and central differencing, respectively. For the full gradient, the mean-squared error is given by

$$\sigma_g^2 \equiv \mathbb{E}[\|g(x) - \nabla\phi(x)\|^2] = \sum_{i=1}^{n} \sigma_{g,e_i}^2. \tag{2.15}$$

With the formulas (2.13), (2.14) in hand, we can now return to the practical implementation of the finite-difference L-BFGS method. We estimate the constants L_i in (2.13) using a second-order difference. Given a direction $p \in \mathbb{R}^n$ where $\|p\| = 1$, we define

$$\Delta(t) = f(x + tp) - 2f(x) + f(x - tp),$$

where t is the second-order differencing interval. The Lipschitz constant L along the direction p can thus be approximated as $L \approx \Delta(t)/t^2$. However, as with the choice of h, we need to be careful in the selection of t, and for this purpose we employ an iterative technique proposed by Moré and Wild [38], whose goal is to find an interval t that satisfies

$$|\Delta(t)| \geq \tau_1 \epsilon_f, \qquad \tau_1 \gg 1 \tag{2.16}$$

$$|f(x \pm tp) - f(x)| \leq \tau_2 \max\{|f(x)|, |f(x \pm tp)|\}, \qquad \tau_2 \in (0, 1). \tag{2.17}$$

These heuristic conditions aim to ensure that t is neither too small nor too large; see [38] for a full description of the Moré-Wild (MW) technique. This technique cannot, however, be guaranteed to find a t that satisfies (2.16), (2.17). To account for this, we developed the following procedure for estimating the constants L_i for forward differencing.

Procedure I. *Adaptive Estimation of L*

1. At the first iteration of the L-BFGS method, invoke the MW procedure to compute t_i, for $i = 1, ..., n$. If such a t_i can be found to satisfy (2.16), (2.17), set $L_i = \max\{10^{-1}, |\Delta(t_i)|/t_i^2\}$ in (2.13). Otherwise, set it to $L_i = 10^{-1}$. Store the L_i in a vector $\mathbf{L}$. To calculate the directional derivative in the Armijo-Wolfe line search, set the second-order differencing interval to $t = \|\mathbf{L}\|/\sqrt{n}$.

2. If at any iteration of the L-BFGS algorithm the line search returns a steplength $\alpha_k < 0.5$, then re-estimate the vector $\mathbf{L}$ by calling the MW procedure as in step 1 at the current iterate x_k.

When the gradient is sufficiently accurate, L-BFGS generates well-scaled directions such that $\alpha_k = 1$ is acceptable. Thus, the occurrence of $\alpha_k < 0.5$ is viewed as an indication that the curvature of the problem may have changed, and should be re-estimated. The function evaluations performed in the estimation of $\mathbf{L}$ will be accounted for in the numerical results presented below.

For central differencing, a practical procedure for estimating M is more involved, and will be discussed in a forthcoming paper [54]. Here, we simply approximate M using changes in the second derivative of the true objective:

$$M = \max\left\{10^{-1}, \frac{\left|p^T\left(\nabla^2\phi\left(x + \hat{h}\frac{p}{\|p\|}\right) - \nabla^2\phi(x)\right)p\right|}{\hat{h}\|p\|^2}\right\} \tag{2.18}$$

where $\hat{h} = \sqrt{\epsilon_M}$. This is an idealized strategy, and its cost is not accounted for in the results below since central differencing does not play a central role in this study.

In addition to the choice of h mentioned above, we have found it important to modify the line search slightly to better handle noise. Following Shi et al. [53], we relax the Armijo condition within the Armijo-Wolfe line search, as follows. Let the superscript j denote the jth iteration of the line search performed from the iterate x_k. The modified Armijo condition is

$$f(x_k + \alpha_k^j p_k) \leq \begin{cases} f(x_k) + c_1\alpha_k^j g(x_k)^T p_k & \text{if } j = 0,\ g(x_k)^T p_k < -\sigma_g\|p_k\| \\ f(x_k) + c_1\alpha_k^j g(x_k)^T p_k + 2\sigma_f & \text{if } j \geq 1,\ g(x_k)^T p_k < -\sigma_g\|p_k\| \\ f(x_k) & \text{if } g(x_k)^T p_k \geq -\sigma_g\|p_k\|. \end{cases} \tag{2.19}$$

Other than this modification to the line search, no other changes are made to the L-BFGS method. We will refer to the resulting method simply as L-BFGS to avoid introducing more acronyms.

We perform tests to compare L-BFGS with forward and central differencing against NEWUOA, both in terms of the achievable solution accuracy and in terms of efficiency (as measured by function evaluations).

2.2.1 Accuracy

We first compare the accuracy achieved by each algorithm, as measured by the optimality gap $\phi(x_k) - \phi^*$. We do so by running NEWUOA until $\rho = \rho_{\text{end}} = 10^{-6}$, and running the finite-difference L-BFGS method until the objective function could not be improved over 5 consecutive iterations. In Figure 2, we report the log-ratio profile

$$\log_2\left(\frac{\phi_{\text{LBFGS}} - \phi^*}{\phi_{\text{NEWUOA}} - \phi^*}\right) \tag{2.20}$$

for $\sigma_f = 10^{-5}$, where ϕ_{LBFGS}, ϕ_{NEWUOA} denote the lowest objective achieved by each method. (The results are representative of those obtained for $\sigma_f \in \{10^{-1}, 10^{-3}, 10^{-7}\}$.) As in the noiseless case, we removed 4 problems where NEWUOA terminates within the first $2n + 1$ function evaluations for all noise levels, as well as thirteen problems where the algorithms are known to converge to different minimizers without noise.

As seen in Figure 2, NEWUOA achieves higher accuracy in the solution than forward-difference L-BFGS, while central-difference L-BFGS yields far better accuracy than both. It is not surprising that central differencing yields much higher accuracy than forward

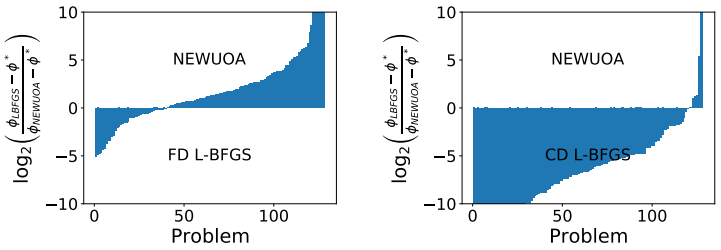

Figure 2: *Accuracy, Noisy Case for* $\sigma_f = 10^{-5}$. Log-ratio optimality gap profiles comparing NEWUOA against forward-difference L-BFGS (left) and central-difference L-BFGS (right).

differencing since the noise level of their gradient approximations is, respectively, $O(\sigma_f^{2/3})$ and $O(\sigma_f^{1/2})$. On the other hand, it is not straightforward to analyze the error contained in the gradient approximation constructed by NEWUOA, and in turn its final accuracy. To try to shed some light into this question, we performed the following experiments.

Recall that NEWUOA by default uses $p = 2n + 1$ interpolation points when constructing the quadratic model of the objective, while forward differencing only uses $n + 1$ points. It is then natural to test the performance of NEWUOA with only $p = n + 2$ interpolation points. The results in Figure 3 show that NEWUOA now lags behind finite-difference L-BFGS, which may be due the fact that the quality of its gradient also depends on the poisedness of the interpolation set [8, 9, 18]. Additional tests, given in Appendix B, suggest that the choice $p = 2n+1$ recommended by Powell strikes the right balance between accuracy in the solution and the speed of algorithm.

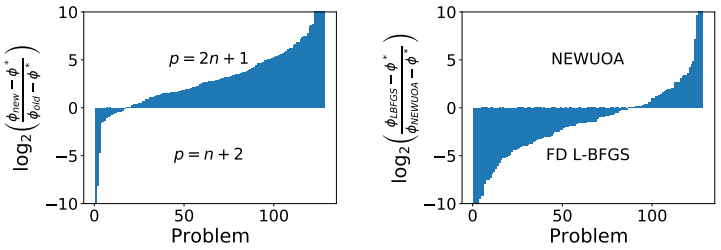

Figure 3: *Accuracy for Non-Default Version of* NEWUOA*, Noisy Case for* $\sigma_f = 10^{-5}$. Log-ratio optimality gap profiles comparing NEWUOA with $p = 2n + 1$ and $p = n + 2$ points (left), and NEWUOA with $p = n + 2$ points against forward-difference L-BFGS (right).

It has been commonly suggested that estimating the Lipschitz constant L_i along each coordinate at the initial point is sufficient for the entire run of a finite-difference method [23, 38]. This, in fact, is not the case in our experiments. We compare in Figure 4 forward-difference L-BFGS with the adaptive Lipschitz estimation given in Procedure I against estimating L only once at the beginning of the run. We see that, in terms of solution accuracy, fixing the Lipschitz constant L at the start does not yield nearly as good accuracy as the

adaptive procedure. This is because that initial estimate is often not a proper bound on the second derivative in the later stages of the run, hence yielding a poor estimate of the finite-difference interval h, and consequently greater error in the gradient approximation. A variety of other strategies for estimating L are discussed in Appendix A.

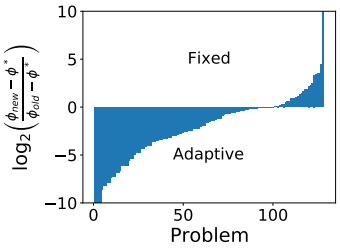

Figure 4: *Accuracy of Fixed and Adaptive Lipschitz Estimation for* L-BFGS*, Noisy Case for* $\sigma_f = 10^{-5}$. Log-ratio optimality gap profiles comparing fixed Lipschitz estimates against adaptive Lipschitz estimation for forward-difference L-BFGS.

2.2.2 Efficiency

To compare the efficiency of different algorithms, we record the number of function evaluations required to reach a particular function value. We use the best solution from NEWUOA (denoted by ϕ_{new}) as a baseline and report the number of function evaluations necessary to achieve

$$\phi(x_k) - \phi_{new} \leq \tau \cdot (\phi(x_0) - \phi_{new}), \tag{2.21}$$

for varied τ. In Figure 5, we report log-ratio profiles comparing the number of function evaluations necessary to satisfy (2.21), i.e.,

$$\log_2\left(\frac{\text{evals}_{\text{LBFGS}}}{\text{evals}_{\text{NEWUOA}}}\right). \tag{2.22}$$

We observe from this figure that NEWUOA is more efficient than forward-difference L-BFGS. The advantage is less significant for $\tau = 10^{-2}$ but becomes pronounced for $\tau = 10^{-6}$, which is to be expected because, as mentioned above, the forward-difference option struggles to achieve as high accuracy as NEWUOA. Central-difference L-BFGS is also less efficient overall than NEWUOA, but becomes more competitive as τ is decreased, which is a product of its ability to converge to a higher accuracy than forward differencing. It is notable that NEWUOA is able to deliver such strong performance without knowledge of the noise level of the function. The adjustment of the two trust-region radii in NEWUOA seems to be quite effective: shrinking the radii fast enough to ensure steady progress, but not so fast as to create models dominated by noise. To our knowledge, there has been no in-depth study of the *practical* behavior of NEWUOA (or similar codes) in the presence of noise; we regard this as an interesting research topic.

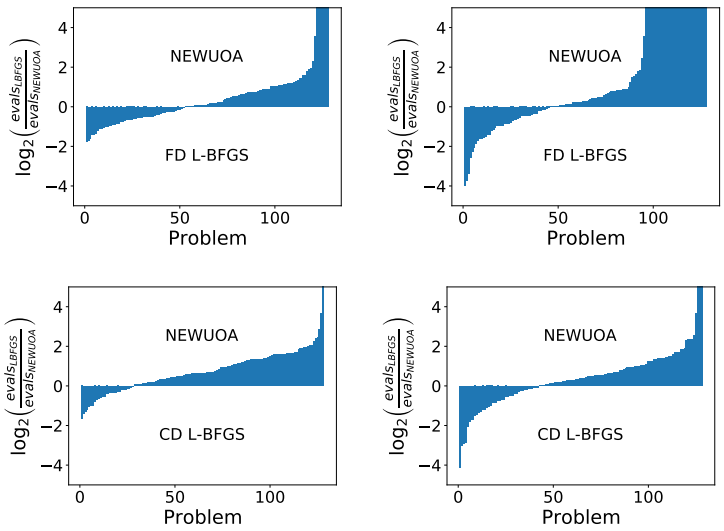

Figure 5: *Efficiency, Noisy Case for* $\sigma_f = 10^{-5}$. Log-ratio profiles for the number of function evaluations to achieve (2.21) for $\tau = 10^{-2}$ (left) and 10^{-6} (right), comparing NEWUOA against forward-difference L-BFGS (top) and central-difference L-BFGS (bottom). These plots are representative of the other noise levels $\sigma_f \in \{10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}\}$.

We also compare the efficiency of NEWUOA with only $p = n + 2$ interpolation points against forward-difference L-BFGS; see Figure 6. We observe that forward-difference L-BFGS is more competitive in this case. More details are given in Appendix B.

One conclusion from these results is that a more sophisticated implementation of finite differences is required to close the performance gap between L-BFGS and NEWUOA. This requires a more accurate, yet affordable, mechanism for estimating the Lipschitz constants in the course of the run. The design of an adaptive procedure of this kind is the subject of a forthcoming paper [54]. It must be able to deal with some of the challenging situations described next.

2.2.3 Commentary

It is well known in computational mathematics that finite-difference methods can be unreliable in the solution of differential equations when knowledge of higher-order derivative estimates is poor. Optimization, however, provides a more benign setting in that a poor choice of h that leads to a bad step can, in many situations, be identified and corrective action can be taken. Gill et al. [23] propose an iterative procedure for estimating h, but we believe that more sophisticated strategies must be developed to improve the reliability of finite-difference DFO methods. The design of such techniques is outside the scope of this paper, but we now give one example illustrating the challenges that arise and some opportunities for addressing them.

Let us consider the use of forward differences for the solution of the `DENSCHNE` problem

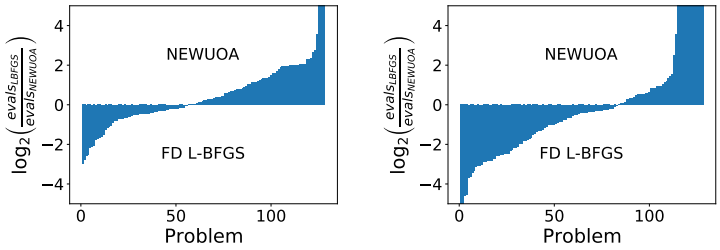

Figure 6: *Efficiency for Non-Default Version of* NEWUOA*, Noisy Case.* Log-ratio profiles for the number of function evaluations to achieve (2.21) for $\tau = 10^{-2}$ (left) and 10^{-6} (right) for $\sigma_f = 10^{-5}$ comparing NEWUOA with $p = n + 2$ interpolation points against forward-difference L-BFGS.

from the `CUTEst` collection, which is given by

$$\min_{x \in \mathbb{R}^3} \phi(x) = x_1^2 + (x_2 + x_2^2)^2 + (-1 + e^{x_3})^2, \quad \text{with } x_0 = (2, 3, -8).$$

Let us employ a different interval for each variable, and let us focus in particular on x_3. The choice of the differencing interval h_3 depends on an estimate L of the second derivative. We have that

$$\frac{\partial \phi}{\partial x_3}(x) = 2e^{x_3}(e^{x_3} - 1), \qquad \frac{\partial^2 \phi}{\partial x_3^2}(x) = 2e^{x_3}(2e^{x_3} - 1).$$

Let us define L to be the value of the second derivative at $x_3 = -8$. This gives $L = \frac{2e^8 - 4}{e^{16}} \approx 6.705 \times 10^{-5}$, which reflects the fact that ϕ is very flat at that point. Suppose that the noise level is $\sigma_f = 10^{-1}$, then from (2.13), we have that $h \approx 20.539$, which is excessively long given that the function contains an exponential term. The forward-difference derivative approximation, $[g(x)]_3 \approx 3.791 \times 10^9$, is entirely wrong; the correct value is $[\nabla \phi(x)]_3 = 2e^{-8}(e^{-8} - 1) \approx -6.707 \times 10^{-4}$. Clearly, the problem is caused by the fact that the second derivative changes rapidly in the interval $[-8, -8 + 20.539]$ and we employed its lowest instead of its largest value. By greatly underestimating L we computed an harmful differencing interval. (We should note in passing that NEWUOA does not have any difficulties with the `DENSCHNE` problem.)

However, the difficulties arising in this problem can be prevented at two levels. First, it should be possible to design an automatic procedure similar to the one in Gill et al. [23] that would not allow for the creation of such a poor estimate of h. In particular, the procedure should estimate L along an interval, not just in a pointwise fashion. Even if such a procedure is unable to take corrective action, monitoring the optimization step can identify difficulties. For the example given above, the fact that $g(x)$ is huge causes the algorithm to compute a very small steplength α_k. This gives a warning signal that should cause an examination of the interval h, and would immediately reveal that the second derivative has varied rapidly in the differencing interval and that L must be recomputed.

It is easy to envision problems where the finite-difference interval h is underestimated, in which case the effect of noise can be damaging. In addition, one must ensure that automatic procedures for estimating and correcting L are not too costly. Taken, all together, these challenges motivate research into more reliable techniques for finite-difference estimation, *in the context of optimization.*

3 Nonlinear Least Squares

In many unconstrained optimization problems, the objective function has a nonlinear least-squares form. Therefore, it is important to pay particular attention to this problem structure in the derivative-free setting. We write the problem as

$$\min_{x\in\mathbb{R}^n} \phi(x) = \tfrac{1}{2}\|\gamma(x)\|^2 = \tfrac{1}{2}\sum_{i=1}^{m} \gamma_i^2(x),$$

where $\gamma : \mathbb{R}^n \to \mathbb{R}^m$ is a smooth function. We assume that the Jacobian matrix $[J(x)]_{ij} = \frac{\partial \gamma_i(x)}{\partial x_j}$ is not available, but the individual residual functions $\gamma_i(x)$ can be computed. More generally, the evaluation of the γ_i may contain noise so that the observed residuals are given by

$$r_i(x) = \gamma_i(x) + \epsilon_i(x), \quad i = 1, \ldots, m,$$

where $\epsilon_i(x)$ models noise as in (2.1). Thus, the minimization of the true objective function ϕ must be performed based on noisy observations $r_i(x)$ that define the observed objective function

$$f(x) = \tfrac{1}{2}\|r(x)\|^2 = \tfrac{1}{2}\sum_{i=1}^{m} r_i^2(x). \tag{3.1}$$

Since noise is incorporated into each residual function, the model of noise is different from additive noise model in the general unconstrained case discussed in the previous section. In particular, the function evaluation $f(x) = \frac{1}{2}\sum_{i=1}^{m} \gamma_i^2(x) + 2\epsilon_i(x)\gamma_i(x) + \epsilon_i^2(x)$ contains both multiplicative and additive components of noise.

Our goal is to study the viability of methods based on finite-difference approximations to the Jacobian. To this end, we employ a classical Levenberg-Marquardt trust-region method where the Jacobian is approximated by differencing, and perform tests comparing it against a state-of-the-art DFO code designed for nonlinear least-squares problems. The rationale behind the selection of codes used in our experiments is discussed next.

Interpolation Based Trust Region Methods.

Wild [55] proposed an interpolation-based trust-region method that creates a quadratic model for each of the residual functions γ_i, and aggregates these models to obtain an overall model of the objective function. This method has been implemented in POUNDERS [39, 55], which has proved to be significantly faster than the standard approach that ignores the nonlinear least-squares structure and simply interpolates values of f to construct a quadratic model. Zhang, Conn and Scheinberg [57] describe a similar method, implemented in DFBOLS, where each residual function is approximated by a quadratic model using $p \in$

$[n+1, (n+1)(n+2)/2]$ interpolation points; the value $p = 2n+1$ being recommended in practice.

More recently, Cartis and Roberts [51] proposed a Gauss-Newton type approach in which an approximation of the Jacobian $J(x_k)$ is computed by linear interpolation using $n+1$ function values at recently generated points $Y_k = \{x_k, y_1, ..., y_n\}$. Here x_k is the current iterate, which generally corresponds to the smallest objective value observed so far. The approximation $\hat{J}(x_k)$ of the true Jacobian $J(x_k)$ is thus obtained by solving the linear equations

$$r(x_k) + \hat{J}(x_k)(y_j - x_k) = r(y_j), \quad j = 1, ..., n. \tag{3.2}$$

The step of the algorithm s_k is defined as an approximate solution of the trust region problem

$$\min_{s \in \mathbb{R}^n} \; m_k(s) = f(x_k) + g_k^T s + \tfrac{1}{2} s^T H_k s \qquad \text{s.t.} \;\; \|s\|_2 \leq \Delta_k,$$

where $g_k = \hat{J}(x_k)^T r(x_k)$ and $H_k = \hat{J}(x_k)^T \hat{J}(x_k)$. The new iterate is given by $x_{k+1} = x_k + s_k$ and the new set Y_{k+1} is updated by removing a point in Y_k and adding x_{k+1}. As in all model-based interpolation methods, it is important to ensure that the interpolation points do not lie on a subspace of $\mathbb{R}^n$ (or close to a subspace). To this end, the algorithm contains a geometry improving technique that spaces out interpolation points, as needed. The implementation described in [51], and referred to as DFO-GN, is reported to be faster than POUNDERS, and scales better with the dimension of the problem. An improved version of DFO-GN is DFO-LS [12], which provides a variety of options and heuristics to accelerate convergence and promote a more accurate solution. The numerical results reported by Roberts et al. [12] indicate that DFO-LS is a state-of-the-art code for DFO least squares, and therefore will be used in our benchmarking.

Finite-Difference Gauss-Newton Method.

One can employ finite differencing to estimate the Jacobian matrix $J(x)$ within any method for nonlinear least squares, and since this is a mature area, there are a number established solvers. We chose LMDER for our experiments, which is part of the `MINPACK` package [35] and is also available in the `scipy` library. We did not employ LMDIF, the finite-difference version of LMDER, because it does not allow the use of different differencing intervals for each of the residual functions $r_i(x)$; we elaborate on this point below. Another code available in `scipy` is TRF [10], but our tests show that LMDER is slightly more efficient in terms of function evaluations, and tends to give higher accuracy in the solution. The code NLS, recently added to the `Galahad` library [24] would provide an interesting alternative. That method, however, includes a tensor to enhance the Gauss-Newton model, and since this may give it an advantage over DFO-LS, we decided to employ the more traditional code LMDER.

In summary, the solvers used in our tests are:

- `LMDER`: A derivative-based Levenberg-Marquardt trust-region algorithm from the `MINPACK` software library [35], where the finite-difference module is user-supplied by us. We

called the code in Python 3.7 through `scipy` version 1.5.3, using the default parameter settings.

- `DFO-LS`: The most recent DFO software developed by Cartis et al. [12] for nonlinear least squares. This method uses linear interpolation to construct an approximation to the Jacobian matrix, which is then used in a Gauss-Newton-type method. We used version 1.0.2 in our experiments, with default settings except that the `model.abs_tol` parameter is set to 0 to avoid early termination

The test problems in our experiments are those used by Moré and Wild [36], which have also been employed by Roberts and Cartis [51] and Zhang et al. [57]. The 53 unconstrained problems in this test set include both zero and nonzero residual problems, with various starting points, and are all small dimensional, with $n \leq 12$. To measure efficiency, we regard m evaluations of individual residual components, say $\{r_i(\cdot)\}_{i=1}^m$, as one function evaluation. These m evaluations may not necessarily be performed at the same point. We also assume that one can compute any component r_i without a full evaluation of the vector $r(x)$. We terminate the algorithms when either: i) the maximum number of function evaluations $(500 \times n)$ is reached, ii) an optimality gap stopping condition, specified below, is triggered; or iii) the default termination criterion of the two codes is satisfied with tolerance of 10^{-8} (this controls the minimum allowed trust region radius).

3.1 Experiments on Noiseless Functions

We first consider the deterministic case corresponding to $\epsilon_i(x) \equiv 0, \forall i$. For LMDER, we estimate the Jacobian $J(x)$ using forward differences. As before, let ϵ_M denote machine precision. We first evaluate $\{r(x + h_j e_j)\}_{j=1}^n$, where

$$h_j = \max\{1, |[x]_j|\}\sqrt{\epsilon_M},$$

and compute the Jacobian estimate as

$$[\hat{J}(x)]_{ij} = \frac{r_i(x + h_j e_j) - r_i(x)}{h_j}.$$

This formula for h_j does not include a term that approximates the size of the second derivative (c.f.(2.13)) because we observed that such a refinement is not crucial in our experiments with noiseless functions, and our goal is to identify the simplest formula for h that makes a method competitive.

As in the previous section, we consider log-ratio profiles to compare the efficiency and accuracy of LMDER and DFO-LS. We record

$$\log_2\left(\frac{\text{evals}_{\text{DFOLS}}}{\text{evals}_{\text{LMDER}}}\right) \quad \text{and} \quad \log_2\left(\frac{\phi_{\text{DFOLS}} - \phi^*}{\phi_{\text{LMDER}} - \phi^*}\right),$$

where ϕ^* is obtained from [51], ϕ_{DFOLS}, ϕ_{LMDER} denote the best function values achieved by the respective methods, and $\text{evals}_{\text{DFOLS}}$, $\text{evals}_{\text{LMDER}}$ denote the number of function evaluations needed to satisfy the termination test

$$\phi(x_k) - \phi^* \leq \tau \max\{1, |\phi^*|\}, \tag{3.3}$$

for various values of τ. (We set $\text{evals}_{\text{DFOLS}}$ and $\text{evals}_{\text{LMDER}}$ to be a very large number if the above condition is not achieved within the maximum number of function evaluations). The results comparing DFO-LS and LMDER are summarized in Figures 7 and 8. The complete table of results is given in Appendix C.2.1.

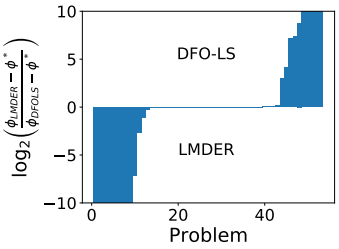

Figure 7: *Accuracy, Noiseless Case.* Log-ratio optimality gap profiles comparing DFO-LS and LMDER for $\epsilon(x) = 0$.

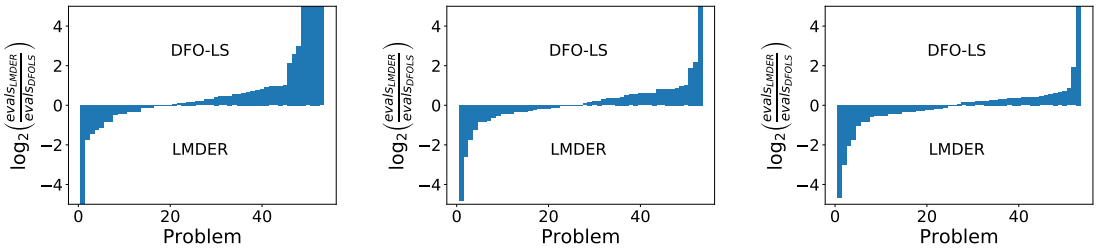

Figure 8: *Efficiency, Noiseless Case.* Log-ratio profiles comparing DFO-LS and LMDER for $\epsilon(x) = 0$. The figures measure number of function evaluations to satisfy (3.3) for $\tau = 10^{-1}$ (left), 10^{-3} (middle), and 10^{-6} (right).

The performance of the two methods appears to be comparable since the area of the shaded regions is similar. This may be surprising since the Gauss-Newton approach seems to be a particularly effective way of designing an interpolation-based trust-region method, requiring only linear interpolation to yield useful second-order information. Cartis and Roberts [51] dismiss finite-difference methods at the outset, and do not provide numerical comparisons with them. However, the tradeoffs of the two methods merit careful consideration. DFO-LS requires only 1 function evaluation per iteration, but its gradient approximation, $\hat{J}(x_k)^T r(x_k)$, is inaccurate until the iterates approach the solution and the trust region has shrunk. In contrast, the finite-difference Levenberg-Marquardt method in LMDER computes quite accurate gradients in this noiseless setting, requiring a much smaller number of iterations, but at a much higher cost per iteration in terms of function evaluations. The tradeoffs of the two methods appear to yield, in the end, similar performance, but we should note that DFO-LS is typically more efficient in the early stages of the optimization, as illustrated in Figure 8 for the low tolerance level $\tau = 10^{-1}$. On the other hand,

the finite-difference approach is more amenable to parallel execution, as mentioned in the previous section.

Let us now consider the linear algebra cost of the two methods. A typical iteration of DFO-LS computes the LU factorization of the matrix $[y_1 - x_k, ..., y_n - x_k]$, and solves the interpolation system (3.2) for n different right hand sides, at a per-iteration cost of $O(mn^2)$ flops, assuming that $m > n$. (DFO-LS offers a number of options, such as regression, which my involve a higher cost, but we did not invoke those options.) The linear algebra cost of the finite-difference version of LMDER described above is $O(mn)$ flops. Therefore, in terms of CPU time, LMDER is faster than DFO-LS in our experiments involving inexpensive function evaluations. This is illustrated in Table 2 for a typical problem (`BROWNAL`) with varying dimensions n and m.

(n, m)	(5,5)	(10,10)	(30,30)	(50,50)	(100,100)	(300,300)	(500,500)
LMDER	0.002	0.003	0.01	0.017	0.046	0.302	0.982
DFO-LS	0.127	0.059	0.762	0.104	0.34	5.404	24.085

Table 2: *Computing Time, Noiseless Case.* CPU time (in seconds) required by DFO-LS and LMDER with forward differencing to satisfy (3.3) for $\tau = 10^{-6}$ on the `BROWNAL` function, as n and m increase.

3.2 Experiments on Noisy Functions

Let us assume that the noise model is the same across all residual functions, i.e., $\epsilon_i(x)$ are i.i.d. for all i. As in Section 2.2, we generate noise independently of x, following a uniform distribution, $\epsilon_i(x) \sim \sigma_f U(-\sqrt{3}, \sqrt{3})$, with noise levels $\sigma_f \in \{10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}\}$.

Differencing will be performed more precisely than in the noiseless case. Following [37], the forward-difference approximation of the Jacobian is defined as

$$[\hat{J}(x_k)]_{ij} = \frac{r_i(x + h_{ij}e_j) - r_i(x)}{h_{ij}}, \quad \text{where } h_{ij} = 8^{1/4}\left(\frac{\sigma_f}{L_{ij}}\right)^{1/2}, \tag{3.4}$$

where L_{ij} is a bound on $|e_j^T \nabla^2 \gamma_i(x) e_j|$ within the interval $[x, x + h_{ij}e_j]$. To estimate L_{ij} for every pair (i, j), and at every iteration, would be impractical, and normally unnecessary. Several strategies can be designed to provide useful information at an acceptable cost. For concreteness, we estimate L_{ij} once at the beginning of the run of LMDER.

More concretely, we compute a different h_{ij} for every residual function γ_i and each coordinate direction e_j at the starting point, and keep h_{ij} constant throughout the run of LMDER. To do so, we compute the Lipschitz estimates $\{L_{ij}\}$ for $i = 1, ..., m, j = 1, ..., n$ by applying the Moré-Wild (MW) procedure [38] described in the previous section to estimate the Lipschitz constant for function γ_i along coordinate directions e_j. If the MW procedure fails, we set $L_{ij} = 1$. The cost of computing the L_{ij}, in terms of function evaluations, is accounted for in the results reported below.

In DFO-LS, we did not employ restarts, and set the `obj_has_noise` option to its default value `False`, which also changes some trust-region-related parameters. We did so for two

reasons. First, restarts introduce randomness, and as Cartis et al. [12] observed, can lead the algorithm to a different minimizer, making comparisons difficult. In addition, restarts are designed to allow the algorithm to make further progress as it reaches the noise level of the function.

Accuracy. We compare the best optimality gap achieved by LMDER and DFO-LS. We run the algorithms using their default parameters (except that we set `model.abs_tol = 0` for DFO-LS) until no further progress could be made. In Figure 3.2, we plot the log-ratio profiles

$$\log_2\left(\frac{\phi_{\text{DFOLS}} - \phi^*}{\phi_{\text{LMDER}} - \phi^*}\right), \tag{3.5}$$

for noise levels $\sigma_f \in \{10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}\}$.

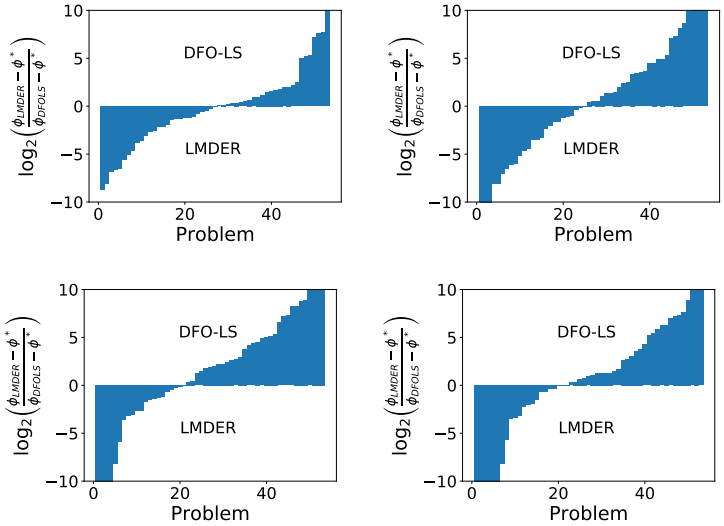

Figure 9: *Accuracy, Noisy Case.* Log-ratio optimality gap profiles comparing DFO-LS and LMDER for $\sigma_f = 10^{-1}$ (upper left), 10^{-3} (upper right), 10^{-5} (bottom left), 10^{-7} (bottom right). The bounds on the second derivatives are kept constant over the optimization process.

We observe that DFO-LS is more accurate than LMDER, which points to the strengths of DFO-LS, since it does not require knowledge of the noise level of the function in its internal logic. However, our implementation of LMDER is not sophisticated. As shown in the previous section, fixing the Lipschitz constant at the start of the finite-difference method is not always a good strategy, and one can ask whether a more sophisticated Lipschitz estimation strategy would close the accuracy gap. To explore this question, we implemented an *idealized strategy* in which the Lipschitz estimation is performed as accurately as possible. For every (i, j), L_{ij} is computed at every iteration, using true function information, via the formula

$$L_{ij} = \frac{|\gamma_i(x + he_j) + \gamma_i(x - he_j) - 2\gamma_i(x)|}{h^2}, \quad \text{with } h = (\epsilon_M)^{1/4}. \tag{3.6}$$

We use this value in (3.4). The results are given in Figure 10, and indicate that LMDER now achieves higher accuracy than DFO-LS on a majority of the problems across most noise levels. This highlights the importance of good Lipschitz estimates in the noisy settings, and suggests that the design of efficient strategies for doing so represent a promising research direction.

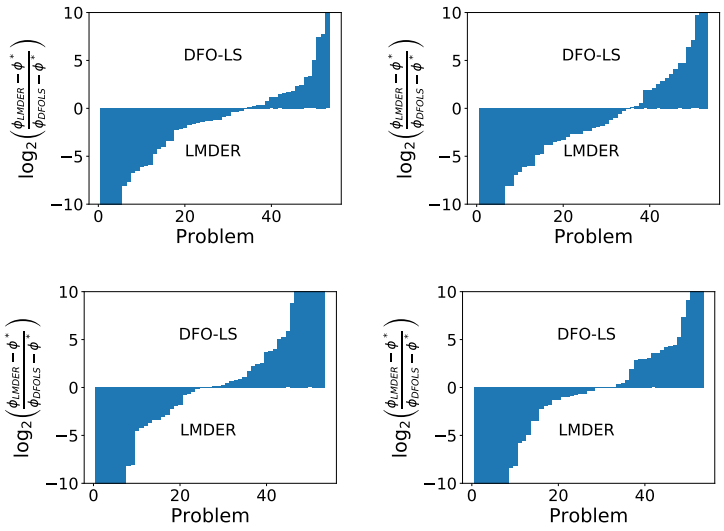

Figure 10: *Accuracy with Idealized Lipschitz Estimation, Noisy Case.* Log-ratio optimality gap profiles comparing DFO-LS and LMDER for $\sigma_f = 10^{-1}$ (upper left), 10^{-3} (upper right), 10^{-5} (bottom left), 10^{-7} (bottom right).

Efficiency. To measure the efficiency of the algorithms in the noisy case, we record the number of function evaluations required to satisfy the termination condition

$$\phi(x_k) - \tilde{\phi}^* \leq \tau(\phi(x_0) - \tilde{\phi}^*), \tag{3.7}$$

where $\tilde{\phi}^*$, denotes the best objective value achieved by the two solvers, for a given noise level. To do so, both solvers were run until they could not make more progress. The differencing interval in LMDER was computed as in the experiments measuring accuracy for the noisy setting, i.e., by employing only the Moré-Wild procedure at the first iteration. In Figure 11, we plot

$$\log_2\left(\frac{\text{evals}_{\text{DFOLS}}}{\text{evals}_{\text{LMDER}}}\right) \tag{3.8}$$

for three values of the tolerance parameter τ and for three levels of noise. We omit the plots for $\sigma_f = 10^{-5}$, which demonstrate similar performance. (When (3.7) cannot be satisfied for a solver within the given budget of $500 \times n$ function evaluations, we set the corresponding value in (3.8) to a very large number.)

There is no clear winner among the two codes used in the experiments reported in Figure 11. For low accuracy ($\tau = 0.1$), LMDER appears to be more efficient, whereas the

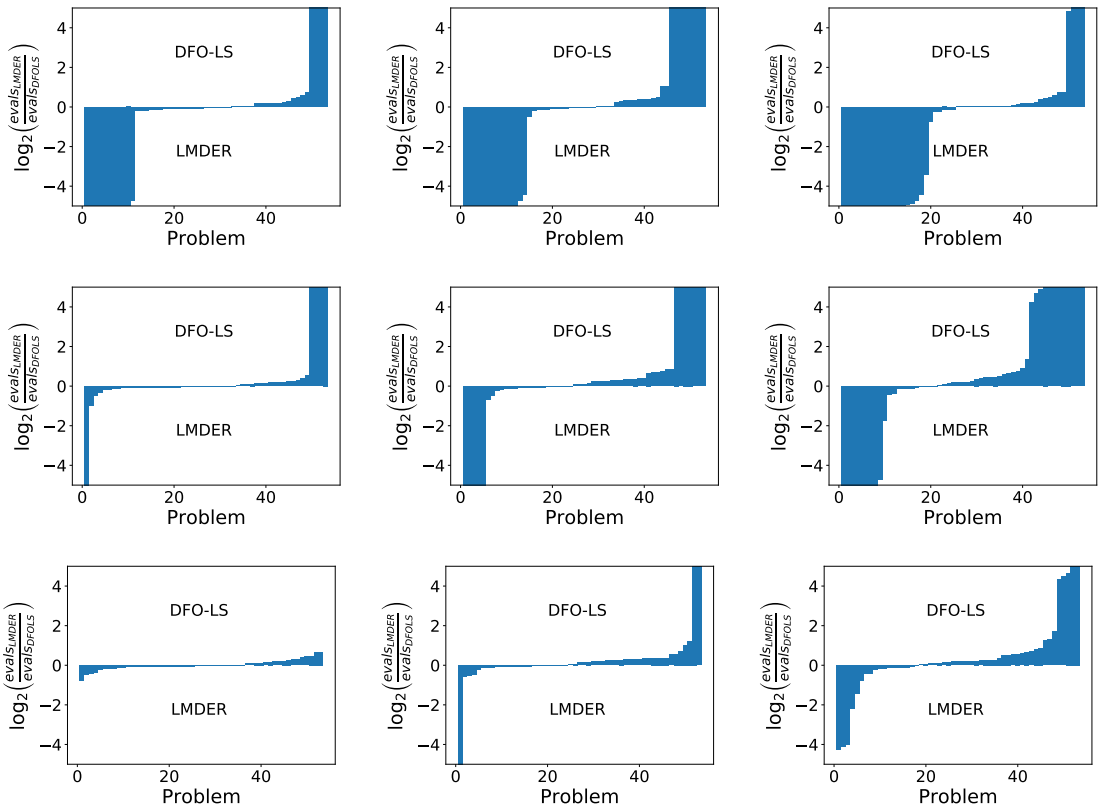

Figure 11: *Efficiency, Noisy Case.* Log-ratio profiles comparing DFO-LS and LMDER for $\sigma_f = 10^{-1}$ (top row), 10^{-3} (middle row), and 10^{-7} (bottom row). The figure measures the number of function evaluations to satisfy (3.3) for $\tau = 10^{-1}$ (left column), 10^{-3} (middle column), and 10^{-6} (right column). Lipschitz constants were estimated only at the start of the LMDER run.

opposite is true for high accuracy ($\tau = 10^{-6}$). We note again that DFO-LS is able to handle different noise levels efficiently and reliably, without knowledge of the noise level or Lipschitz constants. On the other hand, the finite-difference approach is competitive even with a fairly coarse Lipschitz estimation procedure, and perhaps more importantly, it can be incorporated into existing codes (doing so in LMDER required little effort). In other words, in the finite-difference approach to derivative-free optimization, algorithms do not need to be constructed from scratch but can be built as adaptations of existing codes.

3.2.1 Commentary

The nonlinear least squares test problems employed in our experiments are generally not as difficult as some of the unconstrained optimization problems tested in the previous section. This should be taken into consideration when comparing the relative performance of methods in each setting.

It is natural to question the necessity of estimating the Lipschitz constant for each component of each individual residual, as we did in our experiments, since this is affordable only if one can evaluate residual functions individually. One can envision problems for which a much simpler Lipschitz estimation suffices. For example, in data-fitting applications all individual residual functions γ_i may be similar in nature. In this setting, one could use a single Lipschitz constant, say L, across all components and residual functions, especially when the variables are scaled prior to optimization. L could be updated a few times in the course of the optimization process. If the scale of the variables varies signficantly, one can compute Lipschitz constants L_i for each component across all residual functions, requiring the estimation of n Lipschitz constants.

Another possibility is for the variables to be well scaled but the curvature of the residual functions γ_i to vary signficantly. In this case, one could estimate the root mean square of the absolute value of the second derivatives for each component; see Appendix A. This approach notably only requires the estimation of a single constant for each residual function, yielding a total of m Lipschitz constants. This permits the design of a more practical algorithm if the direct estimation of the root mean square is possible.

4 Constrained Optimization

We now consider inequality-constrained nonlinear optimization problems of the form

$$\min_{x\in\mathbb{R}^n} \phi(x) \qquad \text{s.t.} \quad \psi(x) \leq 0, \quad l \leq x \leq u, \tag{4.1}$$

where $\psi : \mathbb{R}^n \to \mathbb{R}^m$ represents a set of m linear or nonlinear constraints, $l, u \in \mathbb{R}^n$, and ϕ and ψ are twice continuously differentiable. We assume that the derivatives of ϕ and ψ are not available, and more generally that we have access only to noisy function evaluations:

$$f(x) = \phi(x) + \epsilon(x), \qquad c_i(x) = \psi_i(x) + \epsilon_i(x). \tag{4.2}$$

(We assume the same noise model for the objective ϵ and each of the constraint functions ϵ_i, for simplicity.) In this section, we compare the numerical performance of an established

DFO method designed to solve problem (4.1) against a standard method for deterministic optimization that approximates the gradients of the objective function and constraints through finite differences.

4.1 Choice of Solvers

The solution of constrained optimization problems using derivative-free methods has not been extensively studied in the literature; see the comprehensive review [30]. There are few established codes for solving problem (4.1) and even fewer for problems that contain both equality and inequality constraints. To our knowledge, the best known software for solving problem (4.1) is COBYLA, developed by Powell [44]. The method implemented in that code constructs linear approximations to ϕ and ψ at every iteration using function interpolation at points placed on a simplex in $\mathbb{R}^n$; it is designed to handle only inequality constraints. A more recent method by Powell, in the spirit of NEWUOA, is LINCOA [45]. It implements an interpolation-based trust-region approach but it can handle only linear constraints. The NOMAD package by Abramson et al. [1] is designed to solve general constrained optimization problems. It is a direct-search method that employs a progressive barrier approach to handle the constraints, and builds quadratic models (or other surrogate functions) as a guide. In the numerical experiments reported in [2], COBYLA outperforms NOMAD, and in [5], NOMAD is not among the best performing methods. Based on these results, we chose COBYLA for our experiments.

There are many production-quality software packages for deterministic constrained optimization where we could implement the finite difference DFO approach. We chose KNITRO because one of the algorithms it offers is a simple sequential quadratic programming (SQP) method that is close in spirit to COBYLA. We did not to employ the interior-point methods offered by KNITRO, which are known to be very powerful techniques for handling inequality constraints, because they may put COBYLA at an algorithmic disadvantage. In the same vein, we did not employ SNOPT because it implements a sophisticated SQP method with many advanced features to improve efficiency and reliability. In short, we selected a simple nonlinear optimization method to more easily identify the strengths and weaknesses of the finite difference approach.

In summary, the codes tested are:

- `COBYLA`. We ran the version of COBYLA maintained in the PDFO package [49]. We set the final trust region radius to 10^{-8} (`rhoend=1e-8`) to observe its asymptotic behavior, particularly in the noiseless case. We ran PDFO version 1.0, and called COBYLA via its Python interface (Python 3.7.7).

- `KNITRO`. We ran Artelys Knitro 12.2 with `alg=4` (an SQP algorithm), `gradopt=2` (forward differencing), and `hessopt=6` (L-BFGS). The choice of the finite difference interval h is described below. In order to make the algorithm as close as possible to COBYLA, we set the memory size of L-BFGS updating to its minimum value, $t = 1$ (`lmsize=1`). For consistency with COBYLA, we disabled the termination test based on the optimality error by setting `opttol=1e-16`, and `findiff_terminate=0`, and

instead terminate when the computed step is less than 10^{-8} (by setting `xtol=1e-8` and `xtol_iters=1`). We called KNITRO via its Python interface.

4.2 Test Problems

As in the previous sections, we used test problems from the `CUTEst` set [25], which were called through the Python interface, `PyCUTEst` version 1.0. We recall from (4.1) that n and m refer to the number of variables and constraints, respectively (excluding bound constraints). We first selected fixed-size problems that have at least one general nonlinear inequality and have no equality constraints, and for which $n \leq 100$ and $m \leq 100$. The properties of the resulting 49 problems are listed in Table 29 in Appendix C.3.1. We also selected 3 variable-size problems with at least one general nonlinear inequality and no equality constraints, and tested them with dimensions up to $n \leq 500$ and $m \leq 500$. These problems are listed in Table 30.

4.3 Experiments on Noiseless Functions

In the first set of experiments, we applied COBYLA and KNITRO to solve the 49 small-scale, fixed-size `CUTEst` problems with exact function evaluations, i.e., with $\epsilon(x) = \epsilon_i(x) \equiv 0$ for all i. As in prior sections, the approximate gradient $g(x) \in \mathbb{R}^n$ of the objective function and Jacobian $\hat{J}(x) \in \mathbb{R}^{m \times n}$ of the constraints are evaluated by finite differencing:

$$[g(x)]_i = \frac{f(x + h_i e_i) - f(x)}{h_i} \tag{4.3}$$

$$[\hat{J}(x)]_{ij} = \frac{c_i(x + h_j e_j) - c_i(x)}{h_j} \tag{4.4}$$

where

$$h_i = \max\{1, |[x]_i|\}\sqrt{\epsilon_M}, \quad i = 1, ..., n. \tag{4.5}$$

Both algorithms were stopped when the number of function evaluations exceed $500 \max(n, m)$, or when the trust-region radius or steplength reaches its lower bound for COBYLA or KNITRO, respectively. Both solvers report feasibility error as the max-norm of constraint violations.

Table 31 reports the results of these tests. A * indicates that the solver hits the limit of function evaluations. We sorted the results in Table 31 into four groups, separated by horizontal lines, according to the following outcomes:

(i) Both solvers converged to a feasible point with approximately the same objective function values.

(ii) The solvers converged to feasible points with different objective function values.

(iii) One of the solvers terminated at an infeasible point.

(iv) Both solvers terminated at infeasible points.

There were 32 problems associated with outcome (i). We can use them to safely compare the performance of the two solvers in terms of accuracy and efficiency. (We comment on outcomes (ii)-(iv) in Appendix C.3.2.)

Given that the two codes achieved feasibility in these 32 problems, we measure accuracy by comparing the best objective function value obtained by each solver with the value ϕ^* obtained by running KNITRO with exact gradients until it could not make further progress. (In all the runs for determining ϕ^*, feasibility error was less than 10^{-10}.) To measure accuracy, we report the ratios

$$\log_2\left(\frac{\max\{\phi_{\text{KNITRO}} - \phi^*, 10^{-8}\}}{\max\{\phi_{\text{COBYLA}} - \phi^*, 10^{-8}\}}\right). \tag{4.6}$$

We compare accuracy up to eight digits because reporting, say, the ratio $10^{-12}/10^{-15}$ would be misleading, given that the accuracy in the constraint violation could have the inverse ratio. The results are presented in Figure 12, which shows that for most problems both solvers were able to achieve eight digits of accuracy; for the remaining problems, KNITRO gave higher accuracy.

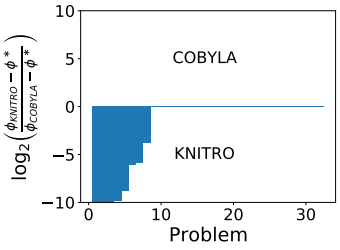

Figure 12: *Accuracy, Noiseless Case.* Log-ratio profiles comparing KNITRO and COBYLA for $\epsilon(x) = \epsilon_i(x) = 0$. The figure plots the ratios (4.6) for problems associated with outcome (i).

To measure efficiency, we conducted a series of experiments using the same subset of 32 problems corresponding to outcome (i). We record the number of function evaluations, $\text{evals}_{\text{KNITRO}}$ and $\text{evals}_{\text{COBYLA}}$, required by the two codes to satisfy the condition

$$\phi_k - \phi^* \leq \tau \cdot \max\{1, |\phi^*|\} \tag{4.7}$$

for various values of τ. (If a solver fails to satisfy this test, we set the number of evaluations to a large value.) Figure 13 plots the ratios

$$\log_2\left(\frac{\text{evals}_{\text{KNITRO}}}{\text{evals}_{\text{COBYLA}}}\right) \tag{4.8}$$

for $\tau \in \{10^{-1}, 10^{-3}, 10^{-6}\}$. Table 32 contains the complete set of results. We observe that for low accuracy, COBYLA is slightly more efficient, while KNITRO becomes significantly more efficient when high accuracy is required.

We should note that employing memory size $t = 1$ in L-BFGS updating yields a very weak quadratic model in the SQP method of KNITRO. We experimented with a memory of size $t = 10$ and observed that the performance of KNITRO improved, particularly in the early iterations of the runs, but not dramatically; see Figure 29 in Appendix C.3.2.

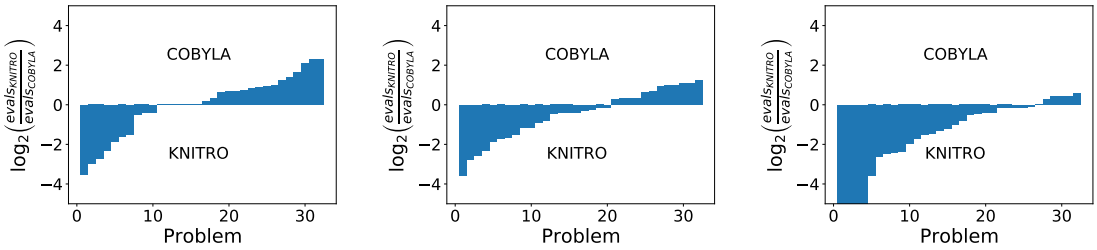

Figure 13: *Efficiency, Noiseless Case.* Log-ratio profiles comparing KNITRO and COBYLA for $\epsilon(x) = \epsilon_i(x) = 0$. The figures measure number of function evaluations to satisfy (4.7) for $\tau = 10^{-1}$ (left), 10^{-3} (middle), and 10^{-6} (right) for outcome (i).

Experiments on Variable-Dimension Problems without Noise. Next, we tested the two codes on the three problems of variable dimensions listed in Table 30, setting $\epsilon(x) = \epsilon_i(x) \equiv 0$. We impose a time limit of an hour; the runs exceeding the time limit are marked with a *T*. Table 3 displays the results. A clear picture emerges from these tests: KNITRO is more reliable than COBYLA in terms of feasibility, tends to achieve a lower objective value, and can solve larger problems within the allotted time. We observed earlier that there is a perception in the DFO literature that finite differences require too many function evaluations, especially as the dimension of the problem increases. The results in Table 3 do not support that concern.

	KNITRO				COBYLA			
problem	ϕ_k	#evals	CPU time	feaserr	ϕ_k	#evals	CPU time	feaserr
SVANBERGN10	15.7315	168	0.078	2.62E-14	26.0000	302	0.349	0.00E+00
SVANBERGN50	82.5819	950	0.538	4.11E-15	136.0000	2140	0.941	0.00E+00
SVANBERGN100	166.1972	1851	1.564	3.22E-15	273.5000	4715	9.754	0.00E+00
SVANBERGN500	835.1869	10054	76.771	1.50E-14	-	-	T	-
READING4N2	-0.0723	20	0.014	0.00E+00	-0.0723	50	0.165	0.00E+00
READING4N50	-0.2685	3556	1.767	8.22E-15	-0.0100	6099	0.123	3.61E+00
READING4N100	-0.2799	9957	16.043	8.44E-15	0.0163	8118	22.609	5.40E-01
READING4N500	-0.2893	88614	1198.368	6.00E-15	-	-	T	-
COSHFUNM3	-0.6614	379	0.152	3.77E-16	-0.6614	730	0.341	0.00E+00
COSHFUNM8	-0.7708	4673	1.424	2.22E-16	-0.7708	12500*	0.324	0.00E+00
COSHFUNM14	-0.7732	21500*	6.366	8.23E-13	-0.7731	21500*	3.673	0.00E+00
COSHFUNM20	-0.7733	30500*	10.792	5.42E-09	-0.7731	30500*	8.430	0.00E+00

Table 3: *Variable-Size Problems, Noiseless Case.* Final objective value ϕ_k, number of function evaluations (#evals), CPU time (in seconds), and final feasibility error (feaserr). A * indicates that the maximum number of function evaluations was reached, and *T* that the time limit (1 hour) was reached.

4.4 Experiments on Noisy Functions

We now inject artificial noise in the evaluation of the objective and constraint functions. We use the same noise model as in the unconstrained setting; i.e., uniform noise sampled i.i.d. given by

$$\epsilon(x), \epsilon_i(x) \sim \sigma_f U(-\sqrt{3}, \sqrt{3}),$$

where $\sigma_f \in \{10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}\}$. We use the same formulas for evaluating the finite-difference gradient (4.3) and Jacobian (4.4), but with a different formula for the finite-difference interval:

$$h_i = \max\{1, |[x]_i|\}\sqrt{\sigma_f}, \quad i = 1, ..., n.$$

We do not include a Lipschitz constant because this formula will suffice for our purposes.

In the first experiment, we ran the two codes with their least stringent termination tests to observe the quality of the final solutions. As in the noiseless case, we set `rhoend=1e-8` for COBYLA and `xtol=1e-8` for KNITRO, and impose a limit of $500\max\{n, m\}$ function evaluations. We tested the 32 `CUTEst` problems for which both solvers converged to the same feasible solution in the noiseless case (outcome (i) above). In Figure 14, we compare the accuracy $\phi(x_k) - \phi_*$ in the true objective given by the code codes, up to eight digits, as in (4.6). If the feasibility violation is large compared to the noise level, i.e. $\|\max\{\psi(x_k), 0\}\|_\infty \geq \sqrt{3}\sqrt{\sigma_f}$, we mark the corresponding run as a failure. We observe from Figure 14 that the performance of the two solvers is comparable, with KNITRO slightly more efficient for low accuracy. Interestingly, for the highest accuracy, $\sigma_f = 10^{-7}$, the performance of the codes is remarkably close (note that the log ratio is nearly zero for about half of the problems). The complete set of results is given in Tables 33-38 in Appendix C.

Next, we compare the efficiency of the two solvers for two levels of accuracy in the objective. Specifically, we record the number of function evaluations required to satisfy

$$\phi(x_k) - \tilde{\phi}_* \leq \tau(\phi(x_0) - \tilde{\phi}_*) \quad \text{and} \quad \|\max\{\psi(x_k), 0\}\|_\infty \leq \sqrt{3}\sqrt{\sigma_f}, \tag{4.9}$$

for $\tau \in \{10^{-2}, 10^{-6}\}$, where $\tilde{\phi}_*$ is the minimum objective value obtained by the two codes. Figure 15 plots the ratios (4.8), which suggest that it is difficult to choose between the two codes. Therefore, in our tests on noisy problems, an unsophisticated code (KNITRO/SQP) for deterministic nonlinear optimization, using a simple strategy for choosing the finite difference interval, is competitive with COBYLA, a method specifically designed for derivative-free optimization.

5 Final Remarks

Finite-difference approximations are widely employed within numerical analysis, particularly for solving ordinary and partial differential equations. The dangers and limitations of finite-difference methods are therefore also well-documented, such as the difficulty of approximating high-order derivatives accurately on unstructured grids or a failure to adapt to problems that require local mesh refinement.

Optimization, however, arguably provides a more benign setting than solving differential equations as errors do not accumulate; if a poor choice of the finite-difference interval yields

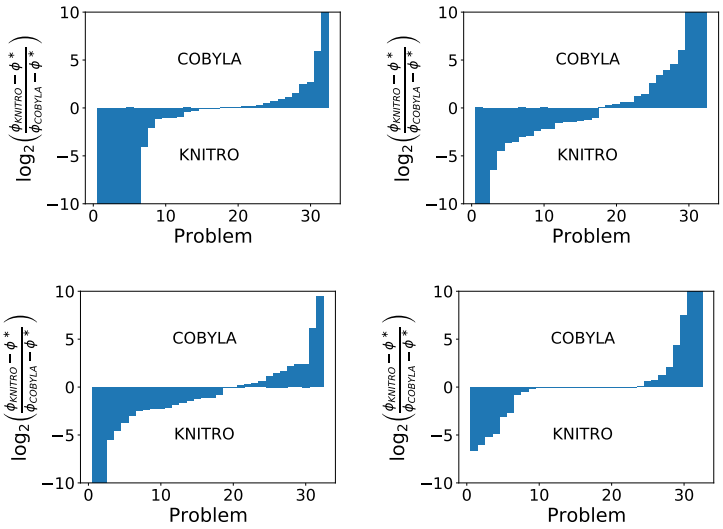

Figure 14: *Accuracy, Noisy Case.* Log-ratio profiles comparing accuracy in the objective by KNITRO and COBYLA, for $\sigma_f = 10^{-1}$ (upper left), 10^{-3} (upper right), 10^{-5} (bottom left), and 10^{-7} (bottom right).

a bad step, it can be detected and improved upon at a later point within the iteration. Two attractive features of finite-difference methods in optimization are the simplicity of incorporating them into existing nonlinear optimization solvers, and their ease of parallelization, in many situations.

Our empirical study has revealed that finite-difference methods can be made competitive, in many cases, against state-of-the-art derivative-free optimization methods. However, algorithmic development and analysis are still necessary to make them sufficiently robust for general-purpose optimization. In addition to more sophisticated procedures for computing the finite difference interval, research is needed in the design of globalization mechanisms such as line searches and trust regions in the noisy setting. We hope that our work draws attention to and provides an empirical basis for further investigation along this important line of inquiry.

Acknowledgements

We are grateful to Richard Byrd, Oliver Zhuoran Liu, and Yuchen Xie for their initial feedback on this work. We also thank Philip Gill, Tammy Kolda, Arnold Neumaier, Michael Saunders, Katya Scheinberg, Luis Vicente and Stefan Wild for their correspondences that led to the design of the experiments in this work.

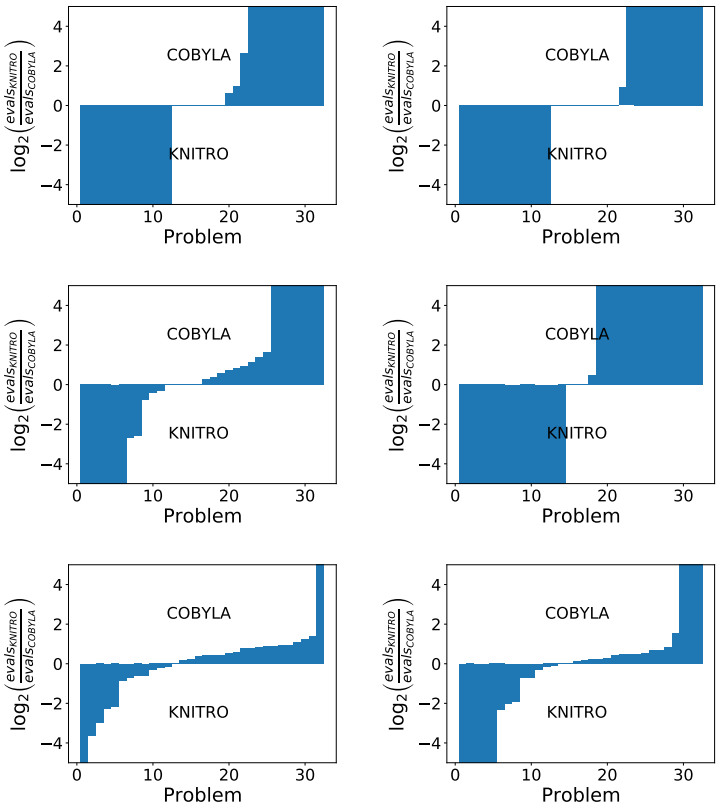

Figure 15: *Efficiency, Noisy Case.* Log-ratio profiles (4.8) comparing KNITRO and COBYLA for $\sigma_f = 10^{-1}$ (top row), 10^{-3} (middle row), 10^{-7} (bottom row). The figure measures the number of function evaluations to satisfy (4.9) for $\tau = 10^{-2}$ (left) and 10^{-6} (right).

References

[1] Mark A Abramson, Charles Audet, G Couture, John E Dennis Jr, Sébastien Le Digabel, and C Tribes. The NOMAD project, 2011.

[2] Charles Audet, Andrew R Conn, Sébastien Le Digabel, and Mathilde Peyrega. A progressive barrier derivative-free trust-region algorithm for constrained optimization. *Computational Optimization and Applications*, 71(2):307–329, 2018.

[3] Charles Audet and John E Dennis Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on optimization*, 17(1):188–217, 2006.

[4] Charles Audet and Warren Hare. Derivative-free and blackbox optimization. 2017.

[5] Florian Augustin and Youssef M Marzouk. NOWPAC: a provably convergent derivative-free nonlinear optimizer with path-augmented constraints. *arXiv preprint arXiv:1403.1931*, 2014.

[6] Afonso S Bandeira, Katya Scheinberg, and Luís Nunes Vicente. Computation of sparse low degree interpolating polynomials and their application to derivative-free optimization. *Mathematical programming*, 134(1):223–257, 2012.

[7] Albert S Berahas, Richard H Byrd, and Jorge Nocedal. Derivative-free optimization of noisy functions via quasi-newton methods. *SIAM Journal on Optimization*, 29(2):965–993, 2019.

[8] Albert S Berahas, Liyuan Cao, Krzysztof Choromanski, and Katya Scheinberg. Linear interpolation gives better gradients than Gaussian smoothing in derivative-free optimization. *arXiv preprint arXiv:1905.13043*, 2019.

[9] Albert S Berahas, Liyuan Cao, Krzysztof Choromanski, and Katya Scheinberg. A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *arXiv preprint arXiv:1905.01332*, 2019.

[10] Mary Ann Branch, Thomas F Coleman, and Yuying Li. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21(1):1–23, 1999.

[11] Richard H Byrd, Jorge Nocedal, and Richard A Waltz. KNITRO: An integrated package for nonlinear optimization. In G. di Pillo and M. Roma, editors, *Large-Scale Nonlinear Optimization*, pages 35–59. Springer, 2006.

[12] Coralia Cartis, Jan Fiala, Benjamin Marteau, and Lindon Roberts. Improving the flexibility and robustness of model-based derivative-free optimization solvers. *ACM Transactions on Mathematical Software (TOMS)*, 45(3):1–41, 2019.

[13] Tony Doungho Choi and Carl T Kelley. Superlinear convergence and implicit filtering. *SIAM Journal on Optimization*, 10(4):1149–1162, 2000.

[14] Andrew R Conn, Katya Scheinberg, and Philippe L Toint. On the convergence of derivative-free methods for unconstrained optimization. *Approximation theory and optimization: tributes to MJD Powell*, pages 83–108, 1997.

[15] Andrew R Conn, Katya Scheinberg, and Philippe L Toint. A derivative free optimization algorithm in practice. In *Proceedings of 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO*, volume 48, page 3, 1998.

[16] Andrew R Conn, Katya Scheinberg, and Luis Vicente. Error estimates and poisedness in multivariate polynomial interpolation. Technical report, IBM T. J. Watson Research Center, 2006.

[17] Andrew R. Conn, Katya Scheinberg, and Luis Vicente. Geometry of interpolation sets in derivative free optimization. *Mathematical Programming, Series A*, 111:141–172, 2007.

[18] Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to derivative-free optimization*, volume 8. SIAM, 2009.

[19] Frank E Curtis and Xiaocun Que. An adaptive gradient sampling algorithm for nonsmooth optimization. *Optimization Methods and Software*, 28(6):1302–1324, 2013.

[20] David Eriksson, David Bindel, and Christine A Shoemaker. pySOT and POAP: An event-driven asynchronous framework for surrogate optimization. *arXiv preprint arXiv:1908.00420*, 2019.

[21] David Eriksson, Michael Pearce, Jacob R Gardner, Ryan Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. *arXiv preprint arXiv:1910.01739*, 2019.

[22] Peter I Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[23] Philip E Gill, Walter Murray, Michael A Saunders, and Margaret H Wright. Computing forward-difference intervals for numerical optimization. *SIAM Journal on Scientific and Statistical Computing*, 4(2):310–321, 1983.

[24] Nicholas IM Gould, Dominique Orban, and Philippe L Toint. GALAHAD, a library of thread-safe fortran 90 packages for large-scale nonlinear optimization. *ACM Trans. Math. Softw.*, 29(4):353–372, 2003.

[25] Nicholas IM Gould, Dominique Orban, and Philippe L Toint. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, 60(3):545–557, 2015.

[26] Genetha A Gray and Tamara G Kolda. Algorithm 856: Appspack 4.0: Asynchronous parallel pattern search for derivative-free optimization. *ACM Transactions on Mathematical Software (TOMS)*, 32(3):485–507, 2006.

[27] Nikolaus Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[28] Carl T Kelley. *Implicit filtering*, volume 23. SIAM, 2011.

[29] Morteza Kimiaei. Line search in noisy unconstrained black box optimization. 2020.

[30] Jeffrey Larson, Matt Menickelly, and Stefan M Wild. Derivative-free optimization methods. *Acta Numerica*, 28:287–404, 2019.

[31] Adrian S Lewis and Michael L Overton. Nonsmooth optimization via quasi-Newton methods. *Mathematical Programming*, 141(1-2):135–163, 2013.

[32] Robert Michael Lewis, Virginia Torczon, and Michael W Trosset. Direct search methods: then and now. *Journal of computational and Applied Mathematics*, 124(1):191–207, 2000.

[33] James N Lyness. Has numerical differentiation a future. In *Proceedings Seventh Manitoba Conference on Numerical Mathematics, Utilitas Mathematica Publishing*, 1977.

[34] José Luis Morales. A numerical study of limited memory BFGS methods, 2002. Applied Mathematics Letters.

[35] Jorge J Moré, Burton S Garbow, and Kenneth E Hillstrom. User guide for MINPACK-1. Technical Report 80–74, Argonne National Laboratory, Argonne, Illinois, USA, 1980.

[36] Jorge J Moré and Stefan M Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1):172–191, 2009.

[37] Jorge J Moré and Stefan M Wild. Estimating computational noise. *SIAM Journal on Scientific Computing*, 33(3):1292–1314, 2011.

[38] Jorge J Moré and Stefan M Wild. Estimating derivatives of noisy simulations. *ACM Transactions on Mathematical Software (TOMS)*, 38(3):19, 2012.

[39] Todd Munson, Jason Sarich, Stefan Wild, Steven Benson, and L Curfman McInnes. Tao 2.0 users manual. *Mathematics and Computer Science Division, Argonne National Laboratory (July 2012)*, 2012.

[40] John A Nelder and Roger Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[41] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

[42] Arnold Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta numerica*, 13(1):271–369, 2004.

[43] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer New York, 2 edition, 1999.

[44] Michael JD Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in optimization and numerical analysis*, pages 51–67. Springer, 1994.

[45] Michael JD Powell. LINearly Constrained Optimization Algorithm. Technical report, Cambridge University, 2005.

[46] Michael JD Powell. The NEWUOA software for unconstrained optimization without derivatives. In *Large-scale nonlinear optimization*, pages 255–297. Springer, 2006.

[47] Michael JD Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, pages 26–46, 2009.

[48] Michael JD Powell. On fast trust region methods for quadratic models with linear constraints. *Mathematical Programming Computation*, 7(3):237–267, 2015.

[49] Tom M. Ragonneau and Zaikun Zhang. PDFO: Cross-platform interfaces for Powell's derivative-free optimization solvers (version 1.0), 2020.

[50] Luis Miguel Rios and Nikolaos V Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, 2013.

[51] Lindon Roberts and Coralia Cartis. A derivative-free Gauss–Newton method. *Mathematical Programming Computation*, 2019.

[52] Nikolaos V Sahinidis and Mohit Tawarmalani. *BARON 7.2.5: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2005.

[53] Hao-Jun Michael Shi, Yuchen Xie, Richard Byrd, and Jorge Nocedal. A noise-tolerant quasi-Newton algorithm for unconstrained optimization. *arXiv preprint arXiv:2010.04352*, 2020.

[54] Hao-Jun Michael Shi, Yuchen Xie, Melody Qiming Xuan, and Jorge Nocedal. Adaptive finite-differencing methods for noisy optimization. 2021. to appear.

[55] Stefan M. Wild. Solving derivative-free nonlinear least squares problems with POUNDERS. In Tamas Terlaky, Miguel F. Anjos, and Shabbir Ahmed, editors, *Advances and Trends in Optimization with Engineering Applications*, pages 529–540. SIAM, 2017.

[56] Margaret H Wright. Direct search methods: Once scorned, now respectable. In *Numerical Analysis 1995 (Proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis)*, pages 191–208. Addison Wesley Longman, 1996.

[57] Hongchao Zhang, Andrew R Conn, and Katya Scheinberg. A derivative-free algorithm for least-squares minimization. *SIAM Journal on Optimization*, 20(6):3555–3576, 2010.

A Numerical Investigation of Lipschitz Estimation

A.1 Investigation of Theoretical Lipschitz Estimates

In Section 2.2, we approximated the bound on the second and third derivative L and M (or the Lipschitz constant of the first and second derivative) along the interval $I = \{x \pm tp : t \in [0, h_0]\}$ for $h_0 > 0$ every time finite-differencing is performed.

For forward-differencing, we employed the Moré and Wild heuristic [38]. The heuristic estimates the bound on the second derivative of a univariate function with noise. Assume $\phi : \mathbb{R} \to \mathbb{R}$ is univariate. If we let $\Delta(t) = f(x+t) - 2f(x) + f(x-t)$, then $t > 0$ must satisfy two conditions:

$$|\Delta(t)| \geq \tau_1 \epsilon_f, \qquad \tau_1 \gg 1 \tag{A.1}$$

$$|f(x \pm t) - f(x)| \leq \tau_2 \max\{|f(x)|, |f(x \pm t)|\}, \qquad \tau_2 \in (0, 1). \tag{A.2}$$

In practice, $\tau_1 = 100$ and $\tau_2 = 0.1$. The first condition ensures that h is sufficiently large such that the second-order difference is not dominated by noise, while the second condition enforces that the difference is not dominated by a particular evaluation, so that there is "equal" contribution from each function evaluation in the finite-difference approximation. If conditions (A.1) and (A.2) are satisfied, then we take $L = \max\{10^{-1}, |\Delta(t)|/t^2\}$.

For central differencing, we used a theoretical estimate based on knowledge of the true Hessian $\nabla^2\phi(x)$:

$$M = \max\left\{10^{-1}, \frac{|p^T(\nabla^2\phi(x + \tilde{h}\frac{p}{\|p\|}) - \nabla^2\phi(x))p|}{\tilde{h}\|p\|^2}\right\} \tag{A.3}$$

where $\tilde{h} = \sqrt{\epsilon_M}$. Since this estimate is ideal, we do not include the cost of evaluating M in the number of function evaluations. In both cases, the maximum is taken to ensure that the bound is strictly bounded away from 0 so that (2.13) and (2.14) remain well-defined.

One may ask whether or not a refined choice of L or M is necessary for finite-difference L-BFGS. To show the impact of Lipschitz estimation on the performance of finite-difference methods, we focus on forward-difference L-BFGS and compare nine different theoretical Lipschitz estimation schemes. The first five consider techniques where L is fixed for the entire run based on information at the initial point. We test this because it is frequently claimed that using an initial estimate of L is sufficient for the entire run; see [23, 38]. The first approach requires no additional information about the function, while the others incorporate information about the Hessian at the current point. The latter four techniques similarly incorporate information about the Hessian but re-estimate L whenever the finite-difference gradient or directional derivative is evaluated.

All of these techniques rely on the assumption that $|D_p^2\phi(x)| \approx |D_p^2\phi(\xi)|$ for some $\xi \in [x, x + h]$. As we will see, both of these approximations that are based on conventional wisdom are challenged in our experiments.

1. Fix $L = 1$. This requires no additional knowledge about the problem at no added cost.

2. Fix $L = \max\{10^{-1}, \|\nabla^2\phi(x_0)\|_2\}$. Similar to fixing $L = 1$, but incorporates knowledge of the initial Hessian.

3. Fix $L = \max\left\{10^{-1}, \frac{1}{n}\sum_{i=1}^n |[\nabla^2\phi(x_0)]_{ii}|\right\}$. This can be obtained by choosing h such that the bound on $\|g(x_0) - \nabla\phi(x_0)\|_1$ at the initial point is minimized.

4. Fix $L = \max\left\{10^{-1}, \frac{1}{\sqrt{n}}\sqrt{\sum_{i=1}^n [\nabla^2\phi(x_0)]_{ii}^2}\right\}$. This can be obtained by choosing h such that the bound on $\|g(x_0) - \nabla\phi(x_0)\|_2^2$ at the initial point is minimized.

5. Fix L to a vector $(\max\{10^{-1}, |[\nabla^2\phi(x_0)]_{ii}|\})_{i=1}^n$ and use the ith component for estimating the ith component of $g(x)$. Uses $\|L\|_2/\sqrt{n}$ when estimating h for the directional derivative in the line search.

6. Evaluate $L = \max\{10^{-1}, \|\nabla^2\phi(x)\|_2\}$ each time finite-differencing is performed.

7. Evaluate $L = \max\left\{10^{-1}, \frac{1}{n}\sum_{i=1}^n |[\nabla^2\phi(x)]_{ii}|\right\}$ each time finite-differencing is performed.

8. Evaluate $L = \max\left\{10^{-1}, \frac{1}{\sqrt{n}}\sqrt{\sum_{i=1}^n [\nabla^2\phi(x)]_{ii}^2}\right\}$ each time finite-differencing is performed.

9. Evaluate $L = \max\left\{10^{-1}, |[\nabla^2\phi(x)]_{ii}|\right\}$ when evaluating $[g(x)]_i$. When the directional derivative along $p \in \mathbb{R}^n$ is computed, evaluates $L = \max\left\{10^{-1}, \frac{|p^T\nabla^2\phi(x)p|}{\|p\|_2^2}\right\}$.

Each method is terminated when it cannot make more progress over 5 consecutive iterations. The optimal value ϕ^* is obtained by running L-BFGS on the non-noisy function until no more progress can be made.

To compare the solution quality between two algorithms, we will use log-ratio profiles as proposed in [34] over the optimality gaps for each algorithm. The log-ratio profiles report

$$\log_2\left(\frac{\phi_{\text{new}} - \phi^*}{\phi_{\text{old}} - \phi^*}\right) \tag{A.4}$$

for each problem plotted in increasing order. The area of the shaded region is representative of the general success of the algorithm.

One may ask whether or not using $L = 1$ is sufficient, without any additional knowledge of the second derivative. As seen in Figure 16, when compared against the other fixed Lipschitz estimation schemes, the difference is not significant. In particular, the additional information from the Hessian does not yield significant benefits over setting $L = 1$ because the bound on the second derivative does not remain valid over the entire run of the algorithm. This may be surprising as it has been commonly suggested that it is sufficient to fix the Lipschitz estimate; see [23, 38]. In fact, we will see that it is more crucial for the algorithm to have a right estimate of L during the later stages of the run than at the beginning of the run in order to achieve high accuracy.

To further support why an adaptive L is important to achieve high accuracy, we compare the fixed L approaches against the adaptive L in Figure 17.

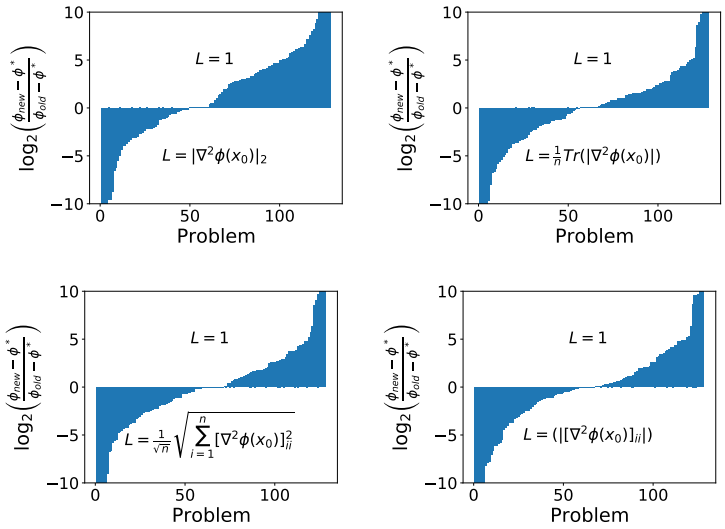

Figure 16: *Accuracy, Noisy Case with* $\sigma_f = 10^{-3}$. Log-ratio optimality gap profiles comparing forward difference L-BFGS with $L = 1$ and other fixed Lipschitz estimation schemes. The noise level is $\sigma_f = 10^{-3}$, but is representative for $\sigma_f \in \{10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}\}$.

As seen in Figure 17, we see that using Lipschitz estimation at every iteration yields much higher accuracy than the alternatives. Upon inspection of individual runs, one can observe many cases where fixing the Lipschitz constant is unstable and inadequate, potentially leading to poor gradient approximations that result in early stagnation of the algorithm. This suggests that it is imperative to adaptively re-estimate L in order to reduce the noise in the gradient and squeeze the best possible accuracy out of forward difference L-BFGS.

In addition, since estimating L for each direction is most competitive out of the adaptive variants as seen in Figure 18, we present the results for a heuristic approach in our main work. However, our investigation reveals that the simple mean or root mean square can provide potentially cheaper alternatives if they can be estimated without knowledge of the componentwise Lipschitz constants.

To our knowledge, there are two weaknesses with the componentwise estimation approach. The first is that for a small subset of problems, it is prone to underestimate the Lipschitz constant, as discussed in Section 2. The second weakness is that the approach, if performed at each iteration and approximated through finite-differencing, is far too expensive. We propose a few possible practical heuristics for handling this in the following section.

A.2 Practical Heuristics for Lipschitz Estimation

Two heuristics were proposed for estimating the Lipschitz constant. Moré and Wild [38] proposed a simple heuristic for estimating the bound on the second derivative of a univariate function with noise, as described in Section 2. Gill, et al. [23] proposed a similar procedure

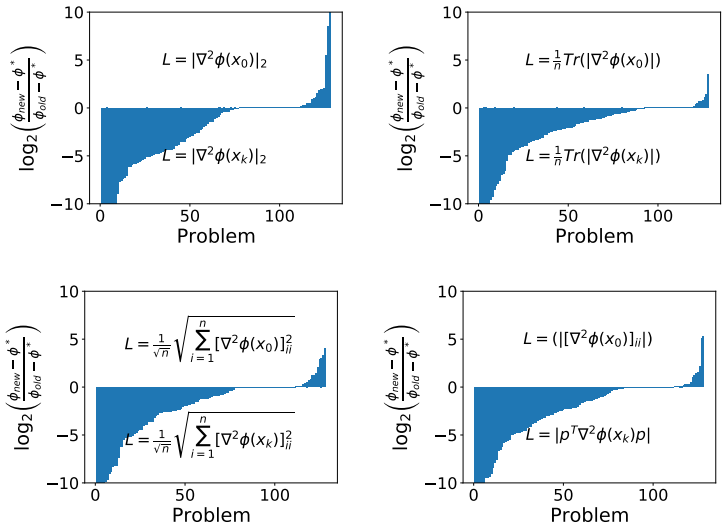

Figure 17: *Accuracy, Noisy Case with* $\sigma_f = 10^{-3}$. Log-ratio optimality gap profiles comparing forward difference L-BFGS with fixed and adaptive Lipschitz estimation schemes. The noise level is $\sigma_f = 10^{-3}$, but is representative for $\sigma_f \in \{10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}\}$.

that instead enforces that the relative cancellation error lies within an interval

$$\frac{4\epsilon_f}{|\Delta(h)|} \in [0.001, 0.1]. \tag{A.5}$$

Note that the lower bound on the interval in (A.5) corresponds to (A.1).

Both of these heuristics were proposed with particular initial guesses of h and additional conditions to handle certain cases where the methods can fail. These heuristics were employed only at the beginning of the iteration for each variable component, then remains fixed for the entire run.

The only work to our knowledge that employs re-estimation of the Lipschitz constant for finite-differencing derivative-free optimization is Berahas, et al. [7]. In his work, the Lipschitz constant is re-estimated whenever the line search fails and the recovery procedure is triggered. Similarly, we will use the Moré and Wild heuristic to estimate the Lipschitz constant, but only re-estimate the Lipschitz constant when $\alpha < 0.5$ after the first iteration, as described in Procedure I in Section 2. We compare two variants of the Lipschitz estimation procedure:

1. `Component MW`: We use the Moré and Wild heuristic to estimate the Lipschitz constant with respect to each component. This is performed at the first iteration and subsequent iterations for line search methods when the line search from the prior iteration gives a steplength $\alpha_k < 0.5$. When estimating the directional derivative along the search direction p_k, we use the root mean square of the component-wise estimates.

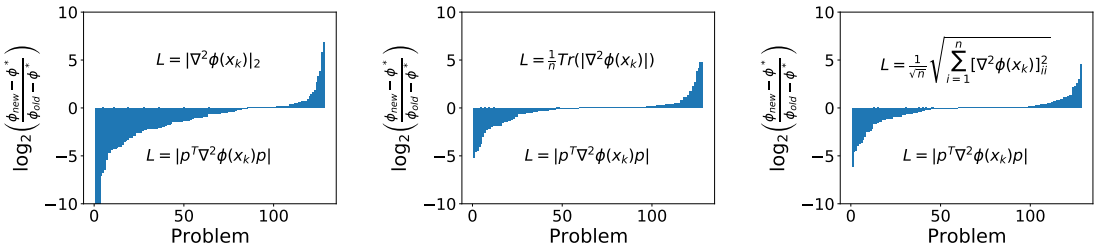

Figure 18: *Accuracy, Noisy Case with* $\sigma_f = 10^{-3}$. Log-ratio optimality gap profiles comparing forward difference L-BFGS with fixed and adaptive Lipschitz estimation schemes. The noise level is $\sigma_f = 10^{-3}$, but is representative for $\sigma_f \in \{10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}\}$.

2. `Random MW`: We use the Moré and Wild heuristic to estimate the Lipschitz constant along a random direction sampled uniformly from a sphere, i.e., $p \sim S(0, I)$. This is performed at the first iteration and subsequent iterations for line search methods when the line search from the prior iteration gives a steplength $\alpha_k < 0.5$.

We compare both Moré and Wild heuristics against NEWUOA and theoretical componentwise Lipschitz estimation in Figures 19, 20, and 21. In general, `Component MW` gives a better solution than `Random MW`. When we compare these methods against NEWUOA in Figure 21, the performance of the methods are relatively the same as the theoretical Lipschitz estimates.

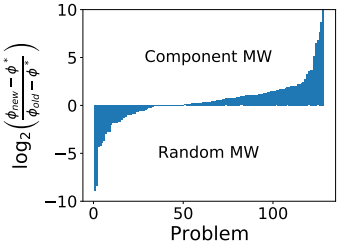

Figure 19: *Accuracy, Noisy Case with* $\sigma_f = 10^{-3}$. Log-ratio optimality gap profiles comparing forward difference L-BFGS with `component MW` and `random MW` Lipschitz estimation schemes.

B Investigation of Parameters for NEWUOA

In this section, we empirically investigate the influence of the parameters for NEWUOA for the noisy setting. Although these parameters have been optimized for the noiseless setting, it is not generally known how these parameters may impact its performance on noisy unconstrained problems. By default, NEWUOA employs $p = 2n + 1$ interpolation

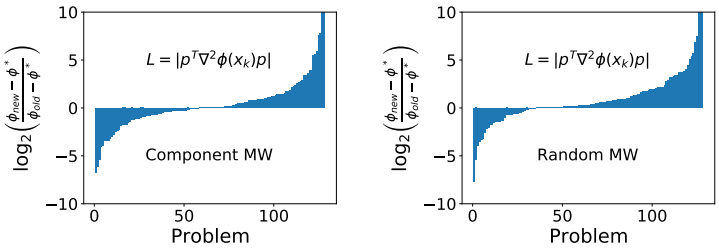

Figure 20: *Accuracy, Noisy Case with* $\sigma_f = 10^{-3}$. Log-ratio optimality gap profiles comparing forward difference L-BFGS with theoretical componentwise Lipschitz estimates and the Moré and Wild Lipschitz estimation schemes.

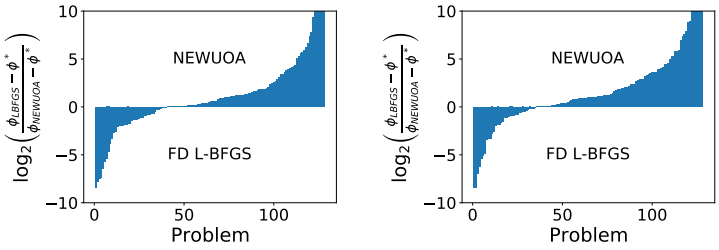

Figure 21: *Accuracy, Noisy Case with* $\sigma_f = 10^{-3}$. Log-ratio optimality gap profiles comparing NEWUOA against forward difference L-BFGS with the Moré and Wild Lipschitz estimation schemes. We compare L-BFGS with `Component MW` (left) and `Random MW` (right).

points when constructing the quadratic model at each iteration with an initial trust region radius of $\rho_{\text{beg}} = 1$ and a final trust region radius of $\rho_{\text{end}} = 10^{-6}$.

B.1 Number of Interpolation Points

In Section 2, we observed that NEWUOA is able to converge to a better quality neighborhood than L-BFGS with forward differencing, but inferior to central differencing. Since NEWUOA by default employs $p = 2n+1$ points, it is natural to both ask if: (1) decreasing the number of interpolation points would yield a less accurate quadratic model, with a potentially less accurate gradient; and (2) increasing the number of points used in the interpolation may improve the accuracy of the solution to be competitive with central differencing.

To do this, we run NEWUOA with $p = n + 2$ and $p = \min\left\{3n + 1, \frac{(n+1)(n+2)}{2}\right\}$ interpolation points and compare their optimality gaps and number of function evaluations. In Figures 22 and 23, we report the log-ratio profiles for the objective function and function evaluations when comparing NEWUOA with $p = n + 2$ and $p = 2n + 1$ points. We see that with higher noise levels, NEWUOA with $p = n + 2$ tends to terminate earlier, while for lower noise levels, it is less efficient than NEWUOA with $p = 2n + 1$ points. In terms of solution quality, NEWUOA with $p = 2n + 1$ is able to converge to a higher accuracy in general than

NEWUOA with $p = n + 2$ points. This is similarly seen when we compare $p = 3n + 1$ and $p = 2n + 1$ points in Figures 24 and 25.

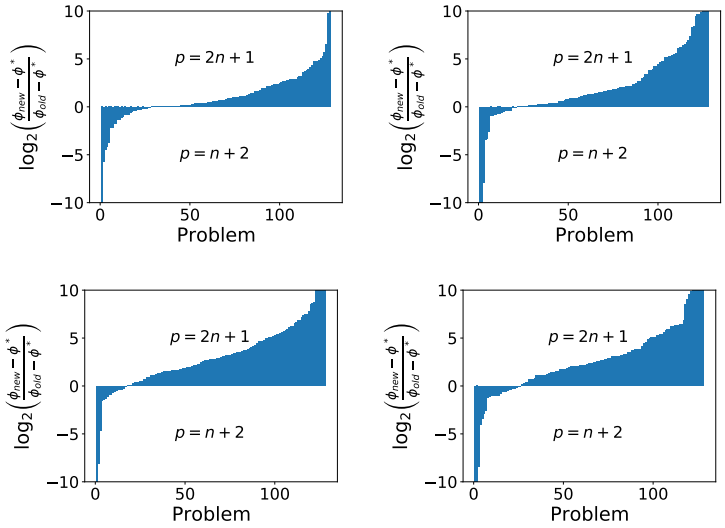

Figure 22: Noisy Case. Log-ratio optimality gap profiles comparing NEWUOA with $p = 2n+1$ and $p = n+2$ points. The noise levels are $\sigma_f = 10^{-1}$ (top left), $\sigma_f = 10^{-3}$ (top right), 10^{-5} (bottom left), and 10^{-7} (bottom right).

When we compare NEWUOA with $p = n + 2$ points against forward difference L-BFGS, we see that L-BFGS is now competitive against NEWUOA, unlike when NEWUOA employed $p = 2n + 1$ points; see Figure 26. However, when we increase the number of points to $p = 3n + 1$, NEWUOA is still not competitive against central differencing, as seen in Figure 27.

B.2 Trust Region Radius

One can also ask if decreasing the final trust region radius could allow NEWUOA to converge to a better solution. To test this, we change $\rho_{\text{end}} = 10^{-12}$ and compare against the default $\rho_{\text{end}} = 10^{-6}$ in Figure 28. From our experiments, changing the final trust region radius makes almost no difference on the solution quality and efficiency.

C Complete Numerical Results

In this appendix, we present our complete numerical results.

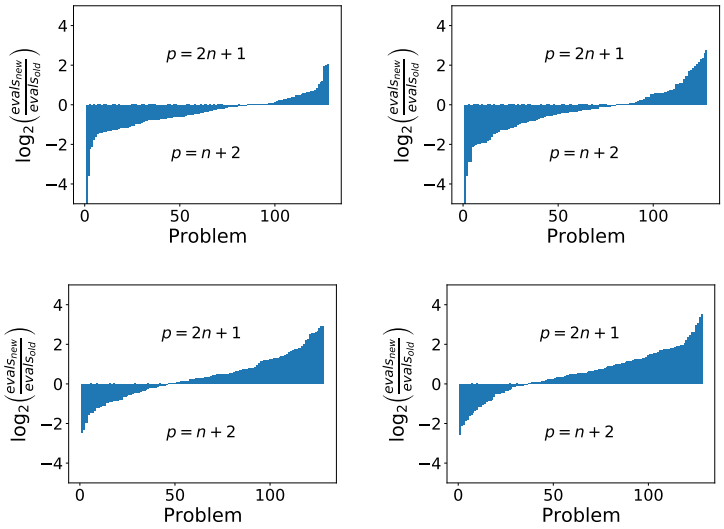

Figure 23: Noisy Case. Log-ratio function evaluation profiles comparing NEWUOA with $p = 2n + 1$ and $p = n + 2$ points. The noise levels are $\sigma_f = 10^{-1}$ (top left), $\sigma_f = 10^{-3}$ (top right), 10^{-5} (bottom left), and 10^{-7} (bottom right).

C.1 Unconstrained Optimization

C.1.1 Noiseless Functions

In this section, we list the number of function evaluations (#feval), ratio between number of function evaluations to the number of function evaluations taken by NEWUOA (RATIO), CPU time (CPU), and optimality gap ($\phi(x) - \phi^*$) for each problem instance in Tables 4-8. Function evaluations marked with a * denote cases where the algorithm reached the maximum number of function evaluations. Instances where NEWUOA samples a point that satisfies (2.7) within the initial $2n+1$ evaluations are denoted by **. Problems marked with a ‡ denote cases where NEWUOA and L-BFGS converge to different minimizers. Optimality gaps marked with a † denote failures of the algorithm to converge to a valid solution satisfying (2.7).

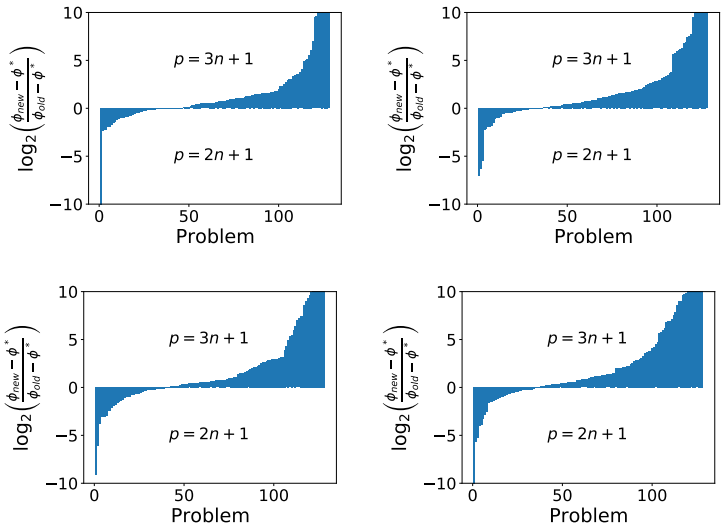

Figure 24: Noisy Case. Log-ratio optimality gap profiles comparing NEWUOA with $p = 3n+1$ and $p = 2n + 1$ points. The noise levels are $\sigma_f = 10^{-1}$ (top left), $\sigma_f = 10^{-3}$ (top right), 10^{-5} (bottom left), and 10^{-7} (bottom right).

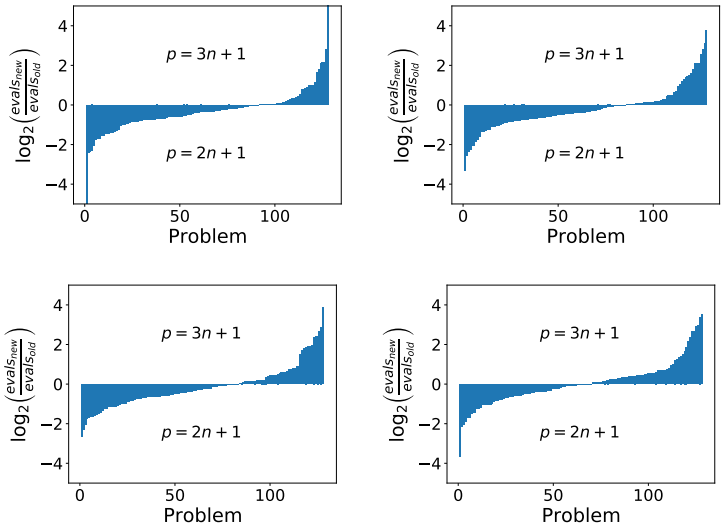

Figure 25: Noisy Case. Log-ratio function evaluation profiles comparing NEWUOA with $p = 3n + 1$ and $p = 2n + 1$ points. The noise levels are $\sigma_f = 10^{-1}$ (top left), $\sigma_f = 10^{-3}$ (top right), 10^{-5} (bottom left), and 10^{-7} (bottom right).

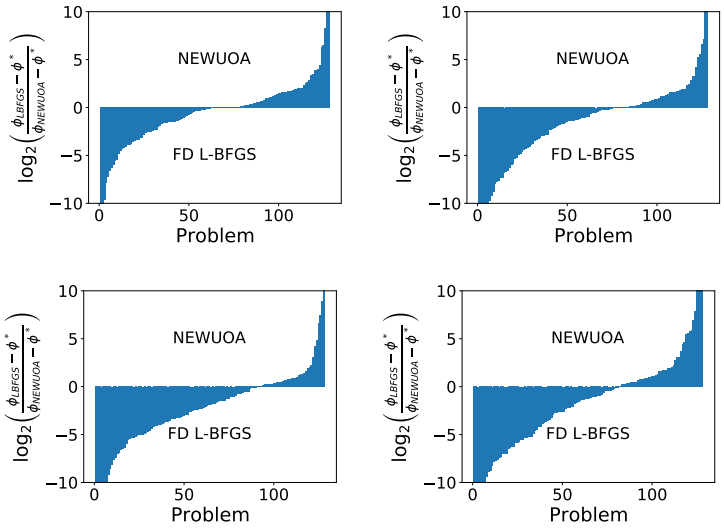

Figure 26: Noisy Case. Log-ratio optimality gap profiles comparing forward difference L-BFGS against NEWUOA with $p = n + 2$ points. The noise levels are $\sigma_f = 10^{-1}$ (top left), $\sigma_f = 10^{-3}$ (top right), 10^{-5} (bottom left), and 10^{-7} (bottom right).

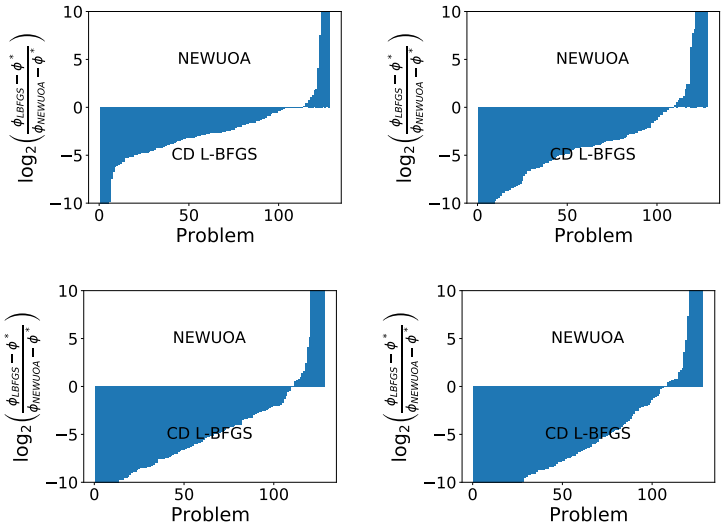

Figure 27: Noisy Case. Log-ratio optimality gap profiles comparing central difference L-BFGS against NEWUOA with $p = 3n + 1$ points. The noise levels are $\sigma_f = 10^{-1}$ (top left), $\sigma_f = 10^{-3}$ (top right), 10^{-5} (bottom left), and 10^{-7} (bottom right).

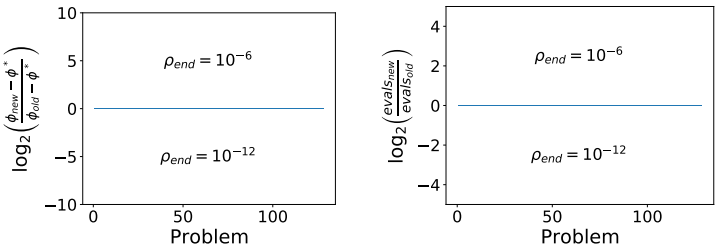

Figure 28: Noisy Case with $\sigma_f = 10^{-5}$. Log-ratio optimality gap profiles comparing NEWUOA with $\rho_{\text{end}} = 10^{-6}$ and 10^{-12}.

		NEWUOA				FD L-BFGS				CD L-BFGS			
Problem	n	#feval	RATIO	CPU	$\phi(x)-\phi^*$	#feval	RATIO	CPU	$\phi(x)-\phi^*$	#feval	RATIO	CPU	$\phi(x)-\phi^*$
AIRCRFTB	5	661	1.000	0.1	6.963e−7	296	0.448	0.02	6.518e−7	536	0.811	0.03	8.288e−7
ALLINITU	4	41	1.000	0.09	3.073e−6	58	1.415	0.01	4.240e−6	103	2.512	0.01	4.240e−6
ARWHEAD	100	201**	1.000	0.09	0.000	1130	5.622	0.06	2.474e−7	2240	11.144	0.07	2.476e−7
BARD	3	69	1.000	0.09	7.032e−8	102	1.478	0.01	5.244e−7	176	2.551	0.01	5.246e−7
BDQRTIC	100	3682	1.000	1.85	3.731e−4	3178	0.863	0.14	2.923e−4	6308	1.713	0.18	2.914e−4
BIGGS3	3	73	1.000	0.09	4.599e−7	85	1.164	0.01	5.422e−7	152	2.082	0.01	5.423e−7
BIGGS5‡	5	103	1.000	0.08	−3.722e−5	416	4.039	0.04	6.651e−7	767	7.447	0.05	4.321e−7
BIGGS6‡	6	338	1.000	0.08	−1.444e−4	323	0.956	0.03	5.357e−7	593	1.754	0.03	5.654e−7
BOX2	2	2**	1.000	0.08	5.847e−19	57	28.500	0.01	9.947e−7	95	47.500	0.01	9.947e−7
BOX3	3	2**	1.000	0.08	5.847e−19	52	26.000	0.01	7.371e−7	91	45.500	0.01	7.371e−7
BRKMCC	2	10	1.000	0.08	1.440e−7	32	3.200	0.0	4.066e−9	51	5.100	0.0	4.088e−9
BROWNAL	10	917	1.000	0.11	9.998e−7	177	0.193	0.01	1.707e−7	331	0.361	0.01	1.707e−7
BROWNAL	100	18069	1.000	9.53	9.993e−7	1039	0.058	0.07	1.040e−7	2051	0.114	0.1	1.047e−7
BROWNAL	200	100000*	1.000	207.31	3.184e−6	823	0.008	0.13	8.254e−7	1626	0.016	0.16	8.243e−7
BROWNDEN	4	140	1.000	0.09	2.900e−2	131	0.936	0.01	1.448e−2	225	1.607	0.01	1.449e−2
CLIFF	2	84	1.000	0.09	5.220e−7	240	2.857	0.02	3.101e−7	365	4.345	0.02	2.076e−7
CRAGGLVY	4	182	1.000	0.09	8.761e−7	135	0.742	0.01	5.854e−7	244	1.341	0.01	5.854e−7
CRAGGLVY	10	511	1.000	0.1	1.650e−6	482	0.943	0.03	1.176e−6	910	1.781	0.04	1.168e−6
CRAGGLVY	50	2670	1.000	0.45	1.523e−5	2301	0.862	0.1	1.289e−5	4544	1.702	0.14	1.292e−5
CRAGGLVY	100	5679	1.000	2.92	3.219e−5	4503	0.793	0.21	1.932e−5	8946	1.575	0.29	1.934e−5
CUBE	2	173	1.000	0.08	8.157e−7	114	0.659	0.01	3.107e−7	190	1.098	0.01	3.036e−7
DENSCHNA	2	22	1.000	0.09	2.893e−10	38	1.727	0.01	9.517e−10	64	2.909	0.01	9.521e−10
DENSCHNB	2	21	1.000	0.09	9.135e−8	25	1.190	0.0	3.867e−11	42	2.000	0.0	3.877e−11
DENSCHNC	2	66	1.000	0.09	7.333e−8	68	1.030	0.01	8.949e−8	111	1.682	0.01	8.951e−8
DENSCHND	3	312	1.000	0.07	9.794e−7	208	0.667	0.02	9.534e−7	356	1.141	0.02	9.537e−7
DENSCHNE	3	127	1.000	0.09	4.487e−7	141	1.110	0.01	1.599e−7	237	1.866	0.01	1.210e−7
DENSCHNF	2	28	1.000	0.09	5.079e−9	48	1.714	0.01	5.450e−9	77	2.750	0.01	5.468e−9

Table 4: Noiseless Unconstrained `CUTEst` Problems Tested. n is the number of variables.

		NEWUOA				FD L-BFGS				CD L-BFGS			
Problem	n	#feval	RATIO	CPU	$\phi(x)-\phi^*$	#feval	RATIO	CPU	$\phi(x)-\phi^*$	#feval	RATIO	CPU	$\phi(x)-\phi^*$
DIXMAANA	15	596	1.000	0.09	9.909e−7	138	0.232	0.01	1.096e−7	265	0.445	0.01	1.096e−7
DIXMAANA	90	1594	1.000	0.61	9.626e−7	738	0.463	0.03	6.577e−7	1465	0.919	0.04	6.578e−7
DIXMAANA	300	8117	1.000	37.53	9.936e−7	2720	0.335	0.15	2.979e−9	5428	0.669	0.21	2.978e−9
DIXMAANB	15	395	1.000	0.1	8.586e−7	122	0.309	0.01	3.845e−9	233	0.590	0.01	3.845e−9
DIXMAANB	90	1491	1.000	0.56	9.724e−7	647	0.434	0.03	6.536e−10	1283	0.860	0.04	6.537e−10
DIXMAANB	300	6446	1.000	27.48	9.980e−7	2117	0.328	0.15	7.825e−10	4223	0.655	0.15	7.845e−10
DIXMAANC	15	197	1.000	0.08	9.747e−7	158	0.802	0.01	8.216e−7	302	1.533	0.01	8.216e−7
DIXMAANC	90	1611	1.000	0.63	9.914e−7	558	0.346	0.03	1.352e−7	1104	0.685	0.03	1.353e−7
DIXMAANC	300	4643	1.000	18.87	9.964e−7	2120	0.457	0.15	5.674e−7	4227	0.910	0.15	5.672e−7
DIXMAAND	15	251	1.000	0.1	9.036e−7	179	0.713	0.01	4.536e−7	340	1.355	0.01	4.536e−7
DIXMAAND	90	2090	1.000	0.81	9.552e−7	745	0.356	0.04	6.242e−7	1474	0.705	0.04	6.242e−7
DIXMAAND	300	7155	1.000	33.18	9.791e−7	2425	0.339	0.18	3.136e−7	4834	0.676	0.18	3.136e−7
DIXMAANE	15	377	1.000	0.1	7.225e−7	293	0.777	0.02	4.958e−7	564	1.496	0.02	4.959e−7
DIXMAANE	90	3081	1.000	1.28	9.840e−7	3499	1.136	0.15	8.197e−7	6956	2.258	0.2	8.202e−7
DIXMAANE	300	18666	1.000	107.25	9.993e−7	19633	1.052	0.96	5.643e−7	39197	2.100	1.42	6.031e−7
DIXMAANF	15	274	1.000	0.09	9.301e−7	226	0.825	0.01	7.285e−7	434	1.584	0.01	7.286e−7
DIXMAANF	90	3104	1.000	1.33	9.964e−7	2674	0.861	0.11	3.730e−7	5313	1.712	0.15	3.731e−7
DIXMAANF	300	16182	1.000	92.5	9.988e−7	14504	0.896	0.69	9.006e−7	28952	1.789	1.06	9.011e−7
DIXMAANG	15	286	1.000	0.08	8.435e−7	277	0.969	0.01	2.101e−7	533	1.864	0.02	2.102e−7
DIXMAANG	90	3429	1.000	1.47	9.922e−7	2857	0.833	0.12	6.520e−7	5678	1.656	0.16	6.522e−7
DIXMAANG	300	21081	1.000	126.91	9.999e−7	14806	0.702	0.78	9.894e−7	30158	1.431	1.1	8.871e−7
DIXMAANH	15	375	1.000	0.1	8.025e−7	280	0.747	0.02	1.341e−7	537	1.432	0.02	1.342e−7
DIXMAANH	90	3106	1.000	1.27	9.943e−7	2588	0.833	0.11	6.407e−7	5138	1.654	0.15	6.405e−7
DIXMAANH	300	35784	1.000	212.15	9.999e−7	14207	0.397	0.69	7.042e−7	28356	0.792	1.05	6.752e−7
DIXMAANI	15	586	1.000	0.1	9.553e−7	549	0.937	0.03	9.880e−7	1060	1.809	0.04	9.880e−7
DIXMAANI	90	13092	1.000	6.05	9.966e−7	15925	1.216	0.62	9.862e−7	33132	2.531	0.91	9.270e−7
DIXMAANI	300	121834	1.000	774.33	9.999e−7	150107*	1.232	6.82	2.119e−6	150155*	1.232	5.3	1.720e−5
DIXMAANJ	15	563	1.000	0.1	9.736e−7	479	0.851	0.03	1.809e−7	926	1.645	0.03	1.797e−7
DIXMAANJ	90	12417	1.000	6.11	9.923e−7	11508	0.927	0.53	9.628e−7	22883	1.843	0.74	9.706e−7
DIXMAANJ	300	120362	1.000	718.6	9.987e−7	77321	0.642	3.61	9.974e−7	150156*	1.248	5.37	1.028e−6
DIXMAANK	15	559	1.000	0.1	9.181e−7	533	0.953	0.03	3.242e−7	1029	1.841	0.04	3.230e−7
DIXMAANK	90	12601	1.000	5.72	9.969e−7	7463	0.592	0.3	9.902e−7	15018	1.192	0.41	9.893e−7
DIXMAANK	300	80808	1.000	455.59	9.997e−7	23567	0.292	1.12	9.623e−7	47649	0.590	1.74	9.108e−7
DIXMAANL	15	752	1.000	0.1	9.740e−7	502	0.668	0.03	4.740e−7	967	1.286	0.03	4.714e−7
DIXMAANL	90	10608	1.000	4.87	9.970e−7	10323	0.973	0.41	9.519e−7	20517	1.934	0.57	8.891e−7
DIXMAANL	300	150000*	1.000	875.24	3.192e−6	34747	0.232	1.67	9.946e−7	69364	0.462	2.51	9.926e−7

Table 5: Noiseless Unconstrained `CUTEst` Problems Tested. n is the number of variables.

		NEWUOA				FD L-BFGS				CD L-BFGS			
Problem	n	#feval	RATIO	CPU	$\phi(x)-\phi^*$	#feval	RATIO	CPU	$\phi(x)-\phi^*$	#feval	RATIO	CPU	$\phi(x)-\phi^*$
DQRTIC	10	502	1.000	0.1	9.719e−7	258	0.514	0.02	4.439e−7	488	0.972	0.02	4.439e−7
DQRTIC	50	2847	1.000	0.43	8.852e−7	1467	0.515	0.06	7.890e−7	2894	1.017	0.08	7.888e−7
DQRTIC	100	6061	1.000	2.81	9.769e−7	3277	0.541	0.13	5.471e−7	6508	1.074	0.17	5.462e−7
EDENSCH	36	928	1.000	0.14	2.175e−4	577	0.622	0.03	7.341e−5	1131	1.219	0.03	7.341e−5
EIGENALS	6	5**	1.000	0.08	0.000	83	16.600	0.01	7.344e−9	152	30.400	0.01	7.099e−9
EIGENALS	110	55000*	1.000	40.44	1.716e−3	42138	0.766	2.12	9.758e−7	55108*	1.002	2.13	2.142e−5
EIGENBLS‡	6	113	1.000	0.09	−3.033e−4	91	0.805	0.01	9.727e−9	167	1.478	0.01	9.145e−9
EIGENBLS	110	37505	1.000	27.4	9.993e−7	55010*	1.467	2.68	9.280e−4	55088*	1.469	2.1	4.002e−2
EIGENCLS	30	1630	1.000	0.19	9.383e−7	2634	1.616	0.12	9.353e−7	5174	3.174	0.16	8.934e−7
ENGVAL1	2	32	1.000	0.08	6.651e−9	38	1.188	0.0	2.191e−7	60	1.875	0.0	2.191e−7
ENGVAL1	50	2234	1.000	0.38	5.211e−5	585	0.262	0.02	3.074e−5	1148	0.514	0.03	3.074e−5
ENGVAL1	100	1941	1.000	0.92	1.076e−4	1337	0.689	0.07	4.197e−6	2651	1.366	0.07	4.198e−6
EXPFIT	2	42	1.000	0.09	9.588e−7	46	1.095	0.01	1.427e−9	75	1.786	0.0	1.420e−9
FLETCBV3‡	10	1106	1.000	0.11	1.063e−3	261	0.236	0.01	−1.664e−5	521	0.471	0.02	−4.710e−5
FLETCBV3‡	100	50000*	1.000	28.66	1.333e5	19708	0.394	0.87	−3.486e−1	16125	0.323	0.54	−1.107e1
FLETCHBV‡	10	945	1.000	0.09	1.490e5	263	0.278	0.02	−1.278e1	278	0.294	0.01	−5.322e3
FLETCHBV‡	100	50000*	1.000	27.77	1.210e13	18068	0.361	0.8	−6.240e6	20933	0.419	0.67	−7.016e7
FREUROTH	2	55	1.000	0.09	2.214e−5	81	1.473	0.01	3.932e−5	134	2.436	0.01	3.932e−5
FREUROTH	10	367	1.000	0.09	6.524e−4	286	0.779	0.02	1.194e−6	538	1.466	0.02	1.194e−6
FREUROTH	50	5840	1.000	0.86	5.844e−3	1053	0.180	0.04	4.888e−4	2073	0.355	0.06	4.886e−4
FREUROTH	100	2881	1.000	1.44	1.188e−2	1950	0.677	0.09	2.601e−3	3868	1.343	0.12	2.589e−3
GENROSE	5	199	1.000	0.08	9.086e−7	226	1.136	0.02	9.527e−9	411	2.065	0.02	9.367e−9
GENROSE	10	645	1.000	0.1	8.369e−7	896	1.389	0.06	8.050e−7	1614	2.502	0.06	1.032e−7
GENROSE	100	17249	1.000	9.45	9.954e−7	25453	1.476	1.01	8.013e−7	50198*	2.910	1.4	1.939e−5
GULF	3	876	1.000	0.12	6.954e−7	225	0.257	0.03	7.913e−8	392	0.447	0.04	6.048e−7
HAIRY	2	261	1.000	0.09	1.214e−6	86	0.330	0.01	2.209e−7	149	0.571	0.01	2.996e−7
HELIX	3	65	1.000	0.09	3.569e−8	149	2.292	0.01	6.812e−7	86	1.323	0.0	8.521e2†
JENSMP‡	2	4**	1.000	0.09	−5.469e2	16	4.000	0.0	0.000	21	5.250	0.0	0.000
KOWOSB	4	150	1.000	0.09	7.089e−7	183	1.220	0.02	2.824e−7	332	2.213	0.02	2.825e−7
MEXHAT	2	46	1.000	0.09	1.036e−2†	203	4.413	0.02	7.395e−7	339	7.370	0.02	4.707e−7

Table 6: Noiseless Unconstrained `CUTEst` Problems Tested. n is the number of variables.

		NEWUOA				FD L-BFGS				CD L-BFGS			
Problem	n	#feval	RATIO	CPU	$\phi(x)-\phi^*$	#feval	RATIO	CPU	$\phi(x)-\phi^*$	#feval	RATIO	CPU	$\phi(x)-\phi^*$
MOREBV	10	379	1.000	0.09	9.458e−7	366	0.966	0.02	6.782e−7	695	1.834	0.03	6.843e−7
MOREBV	50	21488	1.000	3.71	9.988e−7	25029*	1.165	0.98	1.368e−6	25035*	1.165	0.68	6.088e−6
MOREBV	100	11143	1.000	6.29	9.999e−7	15512	1.392	0.59	9.985e−7	31067	2.788	0.82	9.992e−7
NCB20B	21	478	1.000	0.11	4.093e−5	169	0.354	0.01	−4.368e−11	322	0.674	0.01	−4.334e−11
NCB20B	22	627	1.000	0.11	4.361e−5	154	0.246	0.01	2.493e−5	292	0.466	0.01	2.493e−5
NCB20B	50	3345	1.000	0.59	9.936e−5	1465	0.438	0.07	9.906e−5	2892	0.865	0.1	9.912e−5
NCB20B	100	7309	1.000	4.02	1.963e−4	3376	0.462	0.21	1.794e−4	6708	0.918	0.3	1.797e−4
NCB20B	180	11313	1.000	22.07	3.470e−4	4375	0.387	0.35	3.384e−4	8718	0.771	0.56	3.379e−4
NONDIA	10	198	1.000	0.1	4.267e−7	167	0.843	0.01	1.459e−11	298	1.505	0.01	1.053e−11
NONDIA	20	426	1.000	0.1	8.903e−7	341	0.800	0.02	2.571e−10	655	1.538	0.02	2.929e−10
NONDIA	30	569	1.000	0.12	9.936e−7	492	0.865	0.02	3.712e−7	956	1.680	0.03	3.738e−7
NONDIA	50	799	1.000	0.2	9.829e−7	792	0.991	0.03	3.825e−8	1556	1.947	0.04	3.691e−8
NONDIA	90	1302	1.000	0.58	9.192e−7	1301	0.999	0.06	1.233e−7	2574	1.977	0.07	1.188e−7
NONDIA	100	1683	1.000	0.89	9.873e−7	1543	0.917	0.07	1.028e−10	3057	1.816	0.09	1.125e−10
NONDQUAR	100	50000*	1.000	29.28	8.675e−6	50056*	1.001	1.88	1.137e−5	50189*	1.004	1.32	3.611e−5
OSBORNEA‡	5	1094	1.000	0.09	1.204e−5	477	0.436	0.05	8.442e−7	888	0.812	0.06	8.390e−7
OSBORNEB	11	1529	1.000	0.14	9.678e−7	1044	0.683	0.08	9.551e−7	2216	1.449	0.11	6.312e−7
PENALTY1	4	577	1.000	0.09	9.994e−7	868	1.504	0.09	9.134e−7	1425	2.470	0.09	9.856e−7
PENALTY1	10	1332	1.000	0.11	9.944e−7	1523	1.143	0.09	9.903e−7	2942	2.209	0.11	8.873e−7
PENALTY1	50	6082	1.000	0.89	9.985e−7	5773	0.949	0.22	9.994e−7	11797	1.940	0.32	9.652e−7
PENALTY1	100	13215	1.000	6.44	9.969e−7	11175	0.846	0.43	7.672e−7	21381	1.618	0.57	9.919e−7
PFIT1LS‡	3	417	1.000	0.11	2.892e−4	351	0.842	0.03	9.871e−7	403	0.966	0.03	6.795e−7
PFIT2LS‡	3	767	1.000	0.1	1.243e−2	1502*	1.958	0.17	2.818e−4	1509*	1.967	0.11	1.528e−3
PFIT3LS‡	3	907	1.000	0.1	8.228e−2	1502*	1.656	0.17	1.504e−2	1501*	1.655	0.11	3.622e−2
PFIT4LS‡	3	1088	1.000	0.1	2.673e−1	1505*	1.383	0.15	5.424e−2	1508*	1.386	0.1	9.180e−2
QUARTC	25	1004	1.000	0.13	8.886e−7	765	0.762	0.04	4.947e−7	1492	1.486	0.05	4.946e−7
QUARTC	100	6061	1.000	2.82	9.769e−7	3277	0.541	0.13	5.471e−7	6508	1.074	0.16	5.462e−7
SINEVAL	2	254	1.000	0.08	6.706e−9	309	1.217	0.04	3.585e−9	522	2.055	0.04	1.305e−7

Table 7: Noiseless Unconstrained `CUTEst` Problems Tested. n is the number of variables.

		NEWUOA				FD L-BFGS				CD L-BFGS			
Problem	n	#feval	RATIO	CPU	$\phi(x)-\phi^*$	#feval	RATIO	CPU	$\phi(x)-\phi^*$	#feval	RATIO	CPU	$\phi(x)-\phi^*$
SINQUAD	5	125	1.000	0.09	6.682e−6	61	0.488	0.01	3.061e−6	108	0.864	0.01	3.061e−6
SINQUAD	50	3325	1.000	0.55	1.132e−3	795	0.239	0.04	4.489e−6	1562	0.470	0.05	4.553e−6
SINQUAD	100	8318	1.000	4.3	3.985e−3	1545	0.186	0.07	2.755e−3	3062	0.368	0.09	2.758e−3
SISSER	2	24	1.000	0.09	8.489e−7	54	2.250	0.01	5.876e−7	92	3.833	0.01	5.876e−7
SPARSQUR	10	184	1.000	0.09	8.876e−7	184	1.000	0.01	6.336e−7	348	1.891	0.01	6.336e−7
SPARSQUR	50	1645	1.000	0.29	6.775e−7	1046	0.636	0.04	3.744e−7	2065	1.255	0.06	3.744e−7
SPARSQUR	100	2932	1.000	1.37	8.999e−7	2150	0.733	0.09	3.880e−7	4270	1.456	0.12	3.880e−7
TOINTGSS	10	234	1.000	0.09	1.038e−5	24	0.103	0.0	0.000	45	0.192	0.0	0.000
TOINTGSS	50	775	1.000	0.18	9.723e−6	104	0.134	0.0	−3.000e−9	205	0.265	0.01	−3.000e−9
TOINTGSS	100	1199	1.000	0.55	9.778e−6	204	0.170	0.02	4.000e−9	405	0.338	0.01	4.000e−9
TQUARTIC	5	92	1.000	0.09	5.444e−7	85	0.924	0.01	1.078e−8	150	1.630	0.01	1.074e−8
TQUARTIC	10	277	1.000	0.09	9.847e−7	172	0.621	0.01	5.228e−9	325	1.173	0.01	5.313e−9
TQUARTIC	50	5022	1.000	0.75	9.949e−7	646	0.129	0.03	1.681e−8	1265	0.252	0.03	1.725e−8
TQUARTIC	100	20515	1.000	10.51	9.991e−7	1538	0.075	0.07	6.818e−9	3053	0.149	0.08	6.223e−9
TRIDIA	10	186	1.000	0.09	9.019e−7	210	1.129	0.01	5.451e−8	396	2.129	0.01	5.446e−8
TRIDIA	20	446	1.000	0.1	8.062e−7	690	1.547	0.04	8.068e−7	1340	3.004	0.04	8.064e−7
TRIDIA	30	768	1.000	0.13	9.962e−7	1481	1.928	0.07	6.309e−7	2906	3.784	0.08	6.315e−7
TRIDIA	50	1446	1.000	0.3	8.806e−7	3128	2.163	0.15	9.300e−7	6187	4.279	0.19	9.292e−7
TRIDIA	100	3450	1.000	2.11	9.968e−7	8783	2.546	0.32	9.677e−7	17468	5.063	0.43	9.208e−7
WATSON	12	4966	1.000	0.21	9.982e−7	1324	0.267	0.08	9.773e−7	2708	0.545	0.11	9.950e−7
WATSON	31	15500*	1.000	1.11	6.438e−5	13875	0.895	0.65	9.957e−7	15530*	1.002	0.51	5.664e−6
WOODS	4	497	1.000	0.08	1.434e−7	178	0.358	0.02	4.447e−9	312	0.628	0.02	4.353e−9
WOODS	100	50000*	1.000	28.69	1.371e−4	2972	0.059	0.13	7.208e−7	5902	0.118	0.16	6.252e−7
ZANGWIL2	2	12	1.000	0.09	5.085e−7	11	0.917	0.0	−1.000e−10	19	1.583	0.0	−9.999e−11

Table 8: Noiseless Unconstrained `CUTEst` Problems Tested. n is the number of variables.

C.1.2 Noisy Functions

In this section, we list the best optimality gap ($\phi(x) - \phi^*$) and the number of function evaluations (#feval) needed to achieve this for each problem instance, varying the noise level $\sigma_f \in \{10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}\}$, in Tables 9-21. Function evaluations marked with a * denote cases where the algorithm reached the maximum number of function evaluations. Instances where NEWUOA samples a point that satisfies (2.7) within the initial $2d + 1$ evaluations are denoted by **. Problems marked with a ‡ denote cases where NEWUOA and L-BFGS converge to different minimizers.

C.2 Nonlinear Least Squares Problems

C.2.1 Noiseless Functions

In this section, we list the number of function evaluations (#feval), CPU time (CPU), and optimality gap ($\phi(x) - \phi^*$) for each problem instance in Tables 22. Function evaluations marked with a * denote cases where the algorithm reached the maximum number of function evaluations.

C.2.2 Noisy Functions

In this section, we list the best optimality gap ($\phi(x) - \phi^*$) and the number of function evaluations (#feval) needed to achieve this for each problem instance, varying the noise level $\sigma_f \in \{10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}\}$, in Tables 24-27. Function evaluations marked with a * denote cases where the algorithm reached the maximum number of function evaluations.

C.3 Constrained Optimization

C.3.1 Test Problem Summary

C.3.2 Noiseless Functions

In this section, we list the final objective value($\phi(x)$), number of function evaluations (#feval), CPU time (CPU), and feasibility error(feaserr) for each problem instance in Tables 31-32.

We now comment on outcomes (ii)-(iv) mentioned in Section 4.3.

(ii) The solvers converged to feasible points with different objective function values. There are three such problems, for all of which COBYLA terminated due to the limit on the number of function evaluations. We removed the limit on the number of evaluations for COBYLA to see if it converges to a different local solution. For HS67, COBYLA terminates with a final objective function value of -1116.415 after 32606 evaluations, and for CRESC4 it converges to a feasible solution with $f = 1.03523$ after 4706634 evaluations.

(iii) One of the solvers terminated at an infeasible point. There are seven problems for which KNITRO terminated at a feasible point but COBYLA at an infeasible one. For all of these problems, COBYLA hits the limit on the number of function evaluations; it can make further progress in feasibility if the budget of evaluations is increased. For problems HS101,

			NEWUOA		FD L-BFGS		CD L-BFGS	
Problem	n	σ_f	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$
AIRCRFTB	5	1e−1	43	7.257e−1	53	7.277e−1	231	2.623e−1
AIRCRFTB	5	1e−3	70	1.772e−1	715	9.938e−2	904	6.648e−5
AIRCRFTB	5	1e−5	159	1.006e−2	1221	3.355e−4	927	1.042e−5
AIRCRFTB	5	1e−7	317	2.893e−4	1176	4.119e−6	1106	7.139e−12
ALLINITU	4	1e−1	26	1.020e−1	24	2.881	108	1.980e−3
ALLINITU	4	1e−3	38	9.901e−4	97	2.614e−4	189	7.467e−7
ALLINITU	4	1e−5	44	1.337e−6	62	1.721e−5	320	5.606e−9
ALLINITU	4	1e−7	58	1.186e−8	74	1.612e−7	170	3.518e−10
ARWHEAD	100	1e−1	201**	0.000	508	8.433e−2	1022	5.894e−2
ARWHEAD	100	1e−3	201**	0.000	1035	3.525e−2	2037	1.305e−3
ARWHEAD	100	1e−5	201**	0.000	1765	1.925e−4	2444	5.061e−6
ARWHEAD	100	1e−7	201**	0.000	1434	1.346e−5	2660	6.841e−9
BARD	3	1e−1	20	1.665e−1	53	9.347e−1	102	2.119e−3
BARD	3	1e−3	36	5.403e−4	77	1.030e−3	116	1.876e−3
BARD	3	1e−5	47	1.256e−3	294	1.898e−3	310	2.731e−6
BARD	3	1e−7	87	1.685e−7	198	6.076e−8	230	3.753e−9
BDQRTIC	100	1e−1	2283	1.132	2482	5.246	3476	2.697e−1
BDQRTIC	100	1e−3	4736	9.554e−2	2664	1.099e−1	6583	4.061e−3
BDQRTIC	100	1e−5	7755	4.134e−3	6619	3.957e−4	11431	9.216e−7
BDQRTIC	100	1e−7	5083	6.419e−6	6543	6.829e−6	9934	4.423e−9
BIGGS3‡	3	1e−1	8	1.366	160	1.311	168	9.756e−1
BIGGS3‡	3	1e−3	67	2.251e−3	155	6.556e−4	389	1.958e−5
BIGGS3‡	3	1e−5	73	1.776e−6	78	2.858e−6	293	1.380e−6
BIGGS3‡	3	1e−7	85	1.154e−8	95	1.538e−6	389	8.357e−9
BIGGS5‡	5	1e−1	30	1.300	15	1.373	435	1.113e−1
BIGGS5‡	5	1e−3	113	3.932e−2	266	1.181e−1	253	3.142e−2
BIGGS5‡	5	1e−5	176	−4.496e−3	285	1.808e−2	818	8.487e−6
BIGGS‡5	5	1e−7	265	−5.529e−3	580	6.905e−5	1077	2.321e−8
BIGGS6	6	1e−1	30	3.144e−1	121	3.515e−1	283	2.855e−1
BIGGS6	6	1e−3	43	2.937e−1	126	2.897e−1	1549	3.092e−2
BIGGS6	6	1e−5	363	−4.550e−3	586	−3.402e−3	1324	−5.568e−3
BIGGS6	6	1e−7	657	−5.612e−3	1612	−5.608e−3	1778	−5.648e−3
BOX2	2	1e−1	2**	5.847e−19	187	7.590e−2	26	3.168e−1
BOX2	2	1e−3	2**	5.847e−19	66	4.861e−1	387	1.276e−6
BOX2	2	1e−5	2**	5.847e−19	42	4.089e−6	132	1.362e−6
BOX2	2	1e−7	2**	5.847e−19	56	1.849e−6	108	2.693e−8
BOX3	3	1e−1	2**	5.847e−19	28	2.056e−2	55	2.102e−2
BOX3	3	1e−3	2**	5.847e−19	181	1.408e−3	94	4.054e−6
BOX3	3	1e−5	2**	5.847e−19	86	9.697e−6	149	6.273e−7
BOX3	3	1e−7	2**	5.847e−19	67	6.791e−7	370	4.093e−8
BRKMCC	2	1e−1	14	1.160e−2	107	1.085e−1	71	4.542e−3
BRKMCC	2	1e−3	22	8.434e−8	55	3.649e−4	51	1.407e−6
BRKMCC	2	1e−5	10	1.371e−7	45	5.221e−5	62	2.565e−10
BRKMCC	2	1e−7	19	3.325e−10	34	4.961e−8	156	1.995e−10

Table 9: Noisy Unconstrained `CUTEst` Problems Tested. n is the number of variables. σ_f is the standard deviation of the noise.

			NEWUOA		FD L-BFGS		CD L-BFGS	
Problem	n	σ_f	#feval	$\phi(x) - \phi^*$	#feval	$\phi(x) - \phi^*$	#feval	$\phi(x) - \phi^*$
BROWNAL	10	1e−1	46	9.212e−2	53	2.719e−2	384	4.202e−5
BROWNAL	10	1e−3	123	1.232e−2	65	5.759e−5	223	2.969e−5
BROWNAL	10	1e−5	248	3.218e−4	77	3.062e−5	740	1.092e−5
BROWNAL	10	1e−7	458	6.915e−6	77	2.974e−5	821	2.971e−5
BROWNAL	100	1e−1	610	1.164	725	5.055e−2	2664	1.297e−5
BROWNAL	100	1e−3	4084	2.877e−2	725	5.541e−4	3677	1.287e−5
BROWNAL	100	1e−5	7907	3.839e−4	725	1.908e−5	3763	3.096e−8
BROWNAL	100	1e−7	11449	7.999e−5	827	1.336e−5	3540	2.445e−8
BROWNAL	200	1e−1	1514	7.945e−1	821	5.423e−3	7494	8.719e−7
BROWNAL	200	1e−3	11839	3.645e−2	821	4.205e−4	7494	8.234e−7
BROWNAL	200	1e−5	22817	3.574e−4	1833	1.073e−5	5291	9.198e−9
BROWNAL	200	1e−7	26754	5.228e−6	1833	9.284e−7	1626	8.243e−7
BROWNDEN	4	1e−1	219	3.762e−2	151	5.209e−1	327	1.623e−4
BROWNDEN	4	1e−3	163	3.183e−4	358	3.298e−3	276	5.083e−7
BROWNDEN	4	1e−5	239	1.428e−6	351	3.116e−5	751	2.212e−9
BROWNDEN	4	1e−7	224	1.020e−8	159	1.909e−7	292	−4.075e−10
CLIFF	2	1e−1	19	9.252e−2	51	2.902e2	809	5.153e−1
CLIFF	2	1e−3	31	1.389e−3	201	2.902e2	412	3.190e−4
CLIFF	2	1e−5	25	8.089e−4	644	1.027	272	2.209e−4
CLIFF	2	1e−7	68	1.705e−8	407	4.054e−2	335	2.192e−4
CRAGGLVY	4	1e−1	14	8.946e−1	253	7.327e−1	285	4.782e−2
CRAGGLVY	4	1e−3	72	4.048e−3	271	7.551e−3	147	5.397e−4
CRAGGLVY	4	1e−5	173	1.960e−5	262	8.396e−5	338	6.253e−6
CRAGGLVY	4	1e−7	199	1.186e−6	160	3.749e−4	620	3.188e−8
CRAGGLVY	10	1e−1	195	6.511e−1	663	3.023e1	912	2.336e−1
CRAGGLVY	10	1e−3	345	4.337e−2	731	2.444e−2	828	5.647e−4
CRAGGLVY	10	1e−5	573	7.120e−4	716	6.673e−4	1466	4.751e−7
CRAGGLVY	10	1e−7	1066	7.874e−7	1602	5.150e−6	1466	4.121e−9
CRAGGLVY	50	1e−1	1364	1.968	2041	5.834	3529	8.585e−1
CRAGGLVY	50	1e−3	1853	4.989e−2	4908	5.233e−2	4192	9.901e−3
CRAGGLVY	50	1e−5	2312	4.363e−4	3101	9.898e−4	5860	1.281e−5
CRAGGLVY	50	1e−7	3654	2.357e−6	7127	9.978e−6	7120	4.856e−8
CRAGGLVY	100	1e−1	3091	2.974	5275	1.228e1	7272	1.469
CRAGGLVY	100	1e−3	5041	1.878e−1	3918	6.386e−1	7959	1.298e−2
CRAGGLVY	100	1e−5	4955	4.727e−4	6241	5.729e−3	15314	1.573e−5
CRAGGLVY	100	1e−7	7008	5.223e−6	11106	3.053e−5	12688	2.635e−7
CUBE	2	1e−1	26	2.098	34	6.758e−2	65	4.380e−2
CUBE	2	1e−3	68	9.524e−2	43	4.164e−2	346	7.350e−4
CUBE	2	1e−5	135	1.075e−3	40	4.267e−2	213	2.049e−7
CUBE	2	1e−7	145	3.614e−6	141	9.286e−4	563	8.023e−8
DENSCHNA	2	1e−1	12	1.802e−1	31	1.032e−1	72	7.139e−2
DENSCHNA	2	1e−3	18	2.990e−5	66	1.196e−3	387	1.375e−6
DENSCHNA	2	1e−5	22	3.337e−7	73	1.087e−5	82	6.062e−9
DENSCHNA	2	1e−7	22	1.233e−10	44	5.951e−8	151	1.453e−11

Table 10: Noisy Unconstrained `CUTEst` Problems Tested. n is the number of variables. σ_f is the standard deviation of the noise.

			NEWUOA		FD L-BFGS		CD L-BFGS	
Problem	n	σ_f	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$
DENSCHNB	2	1e−1	14	2.282e−2	23	2.369e−1	293	4.982e−4
DENSCHNB	2	1e−3	17	1.807e−5	40	8.803e−4	387	1.432e−6
DENSCHNB	2	1e−5	27	2.693e−7	61	1.017e−6	42	5.983e−11
DENSCHNB	2	1e−7	26	7.582e−9	32	2.194e−8	136	9.498e−12
DENSCHNC	2	1e−1	30	−7.755e−2	18	5.033e−2	281	3.323e−2
DENSCHNC	2	1e−3	62	2.780e−5	80	1.775e−2	219	2.208e−5
DENSCHNC	2	1e−5	55	−1.834e−1	160	6.362e−6	387	1.055e−8
DENSCHNC	2	1e−7	54	−1.834e−1	218	2.200e−6	136	−1.285e−10
DENSCHND	3	1e−1	142	3.571e1	269	3.030e1	228	1.318e−3
DENSCHND	3	1e−3	279	2.258e−3	216	1.288e1	414	4.836e−5
DENSCHND	3	1e−5	216	2.909e−4	240	2.218e−5	326	7.702e−6
DENSCHND	3	1e−7	319	1.084e−6	432	1.318e−7	679	8.619e−8
DENSCHNE	3	1e−1	29	1.570	44	1.003	66	1.038
DENSCHNE	3	1e−3	42	1.003	132	9.994e−1	138	9.993e−1
DENSCHNE	3	1e−5	73	9.999e−1	75	9.993e−1	389	9.993e−1
DENSCHNE	3	1e−7	116	1.080e−8	346	9.993e−1	295	1.651e−10
DENSCHNF	2	1e−1	19	3.023e−3	199	1.909e−2	100	2.349e−3
DENSCHNF	2	1e−3	26	1.388e−5	80	2.299e−4	100	8.045e−6
DENSCHNF	2	1e−5	32	2.913e−8	59	6.314e−7	77	2.843e−8
DENSCHNF	2	1e−7	38	2.403e−9	58	8.740e−10	136	4.048e−11
DIXMAANA	15	1e−1	202	2.011	278	3.989	727	3.807e−3
DIXMAANA	15	1e−3	401	1.671e−2	426	4.111e−3	516	7.512e−6
DIXMAANA	15	1e−5	564	5.826e−5	291	2.041e−5	413	2.145e−8
DIXMAANA	15	1e−7	650	7.482e−7	170	1.288e−6	660	5.438e−11
DIXMAANA	90	1e−1	1357	9.089e−1	1596	1.935e1	4121	1.338e−2
DIXMAANA	90	1e−3	1820	1.564e−2	6783	4.443e−2	3657	3.093e−5
DIXMAANA	90	1e−5	2384	2.123e−4	3552	2.365e−4	3657	1.304e−7
DIXMAANA	90	1e−7	2302	1.072e−6	2031	1.824e−6	4663	1.690e−10
DIXMAANA	300	1e−1	11454	2.772	2091	5.752e1	13555	6.279e−2
DIXMAANA	300	1e−3	12409	3.370e−2	5739	4.750e−2	7272	1.397e−4
DIXMAANA	300	1e−5	21687	2.406e−4	4530	6.880e−4	17895	3.066e−7
DIXMAANA	300	1e−7	18612	3.404e−6	5134	6.485e−6	7860	8.727e−10
DIXMAANB	15	1e−1	199	1.796	136	2.402	278	6.595e−3
DIXMAANB	15	1e−3	384	1.563e−2	700	7.149e−3	270	1.424e−5
DIXMAANB	15	1e−5	389	1.644e−5	351	5.285e−5	505	1.552e−8
DIXMAANB	15	1e−7	373	2.098e−7	274	6.291e−7	491	5.018e−10
DIXMAANB	90	1e−1	1588	1.101	277	1.692e1	1880	2.910e−2
DIXMAANB	90	1e−3	1406	1.029e−2	4163	1.213e−2	1101	5.186e−5
DIXMAANB	90	1e−5	1918	2.728e−4	1568	9.443e−5	2937	2.263e−7
DIXMAANB	90	1e−7	1932	1.523e−6	2282	5.621e−7	2937	1.318e−9
DIXMAANB	300	1e−1	7971	2.504	905	5.660e1	11680	5.947e−2
DIXMAANB	300	1e−3	12927	3.555e−2	11239	9.529e−2	7371	2.935e−4
DIXMAANB	300	1e−5	14656	2.960e−4	8463	2.769e−3	14914	1.849e−6
DIXMAANB	300	1e−7	19729	3.256e−6	5138	2.711e−5	7282	1.165e−9

Table 11: Noisy Unconstrained `CUTEst` Problems Tested. n is the number of variables. σ_f is the standard deviation of the noise.

			NEWUOA		FD L-BFGS		CD L-BFGS	
Problem	n	σ_f	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$
DIXMAANC	15	1e−1	123	3.678e−1	696	9.417e−1	413	4.277e−3
DIXMAANC	15	1e−3	214	1.694e−3	784	9.657e−3	627	7.834e−5
DIXMAANC	15	1e−5	252	1.523e−4	621	3.658e−5	413	2.695e−8
DIXMAANC	15	1e−7	249	2.681e−7	528	4.150e−8	627	1.283e−10
DIXMAANC	90	1e−1	853	1.678	2077	3.626	2649	1.112e−2
DIXMAANC	90	1e−3	2546	1.886e−2	1016	6.657e−2	1880	4.973e−5
DIXMAANC	90	1e−5	2245	1.953e−4	2971	4.662e−4	1880	1.398e−7
DIXMAANC	90	1e−7	2873	2.169e−6	2869	9.707e−6	5905	1.894e−10
DIXMAANC	300	1e−1	5706	2.660	908	2.909e1	12855	5.278e−2
DIXMAANC	300	1e−3	15396	2.200e−2	9990	1.601e−1	13555	1.924e−4
DIXMAANC	300	1e−5	17952	3.248e−4	8167	2.040e−4	9111	2.439e−7
DIXMAANC	300	1e−7	20832	3.234e−6	7261	1.070e−5	6639	9.021e−10
DIXMAAND	15	1e−1	197	2.209	71	2.464	394	1.834e−2
DIXMAAND	15	1e−3	189	1.568e−3	210	3.272e−3	1730	6.885e−6
DIXMAAND	15	1e−5	265	2.640e−5	517	1.972e−4	660	1.423e−7
DIXMAAND	15	1e−7	245	3.120e−7	583	1.876e−6	1730	6.047e−11
DIXMAAND	90	1e−1	1468	1.597	647	4.471	1880	2.476e−2
DIXMAAND	90	1e−3	2428	2.428e−2	2637	6.034e−2	3075	1.075e−4
DIXMAAND	90	1e−5	1511	2.151e−4	5883	2.672e−4	2449	2.126e−7
DIXMAAND	90	1e−7	2339	1.634e−6	2957	1.455e−6	3970	4.576e−10
DIXMAAND	300	1e−1	6087	3.295	1211	2.487e1	11680	6.232e−2
DIXMAAND	300	1e−3	28875	4.695e−2	15937	2.861e−2	13555	2.324e−4
DIXMAAND	300	1e−5	36583	1.968e−4	7259	1.237e−3	6695	8.301e−7
DIXMAAND	300	1e−7	31821	2.379e−6	6958	1.154e−5	7282	1.251e−9
DIXMAANE	15	1e−1	117	2.680	228	2.593	516	4.750e−2
DIXMAANE	15	1e−3	483	6.719e−2	907	2.567e−3	759	9.723e−5
DIXMAANE	15	1e−5	359	6.662e−5	496	4.389e−5	875	3.665e−8
DIXMAANE	15	1e−7	473	5.981e−8	515	4.413e−7	1730	2.043e−10
DIXMAANE	90	1e−1	957	2.077	2228	1.048e1	5905	1.522e−1
DIXMAANE	90	1e−3	2231	5.295e−2	2558	5.724e−2	7854	2.745e−3
DIXMAANE	90	1e−5	2422	1.047e−3	3592	1.658e−3	14178	2.990e−6
DIXMAANE	90	1e−7	3565	3.468e−6	6643	5.020e−6	9472	1.434e−8
DIXMAANE	300	1e−1	5797	6.472	5878	1.576e1	17895	5.082e−1
DIXMAANE	300	1e−3	39321	1.160e−1	11096	1.216e−1	42978	5.040e−3
DIXMAANE	300	1e−5	35575	3.907e−3	16015	3.572e−3	41288	6.960e−5
DIXMAANE	300	1e−7	29189	1.864e−5	27787	3.545e−5	51466	5.905e−8
DIXMAANF	15	1e−1	186	1.292	133	9.307e−1	394	8.259e−2
DIXMAANF	15	1e−3	189	1.051e−2	409	5.640e−3	413	3.139e−5
DIXMAANF	15	1e−5	296	2.329e−5	574	1.101e−4	825	8.220e−8
DIXMAANF	15	1e−7	351	2.811e−7	546	1.188e−6	1730	1.116e−10
DIXMAANF	90	1e−1	1055	1.623	277	6.676	3143	7.567e−2
DIXMAANF	90	1e−3	1855	1.150e−1	2279	3.255e−2	6862	1.656e−3
DIXMAANF	90	1e−5	3117	2.379e−4	4615	5.649e−4	6955	1.941e−6
DIXMAANF	90	1e−7	6296	1.248e−5	4327	6.372e−6	9524	6.810e−9
DIXMAANF	300	1e−1	4652	2.581	904	2.266e1	11680	1.549e−1
DIXMAANF	300	1e−3	10797	1.173e−1	4840	2.153e−1	31414	5.266e−3
DIXMAANF	300	1e−5	41760	2.501e−3	15493	6.550e−3	41516	2.248e−5
DIXMAANF	300	1e−7	17405	3.175e−5	22658	3.595e−5	59871	2.087e−8

Table 12: Noisy Unconstrained `CUTEst` Problems Tested. n is the number of variables. σ_f is the standard deviation of the noise.

			NEWUOA		FD L-BFGS		CD L-BFGS	
Problem	n	σ_f	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$
DIXMAANG	15	1e−1	118	1.301	53	8.760e−1	1730	2.446e−2
DIXMAANG	15	1e−3	219	3.590e−3	532	8.830e−3	1104	4.679e−5
DIXMAANG	15	1e−5	259	1.628e−5	310	9.965e−4	1730	9.830e−8
DIXMAANG	15	1e−7	354	1.419e−7	1235	2.106e−6	1730	3.410e−10
DIXMAANG	90	1e−1	1253	1.630	2284	1.752	4496	9.061e−2
DIXMAANG	90	1e−3	3690	9.785e−2	2011	3.834e−2	6194	1.981e−3
DIXMAANG	90	1e−5	2545	3.136e−4	2579	8.481e−4	6124	7.697e−6
DIXMAANG	90	1e−7	3047	8.060e−6	7112	6.479e−6	8861	7.244e−9
DIXMAANG	300	1e−1	7837	3.509	908	1.259e1	12260	3.429e−1
DIXMAANG	300	1e−3	40449	1.096e−1	9353	2.259e−1	17101	4.560e−3
DIXMAANG	300	1e−5	33893	3.070e−3	8463	2.846e−3	35863	3.047e−5
DIXMAANG	300	1e−7	149998	4.680e−4	21147	6.501e−5	44939	9.867e−8
DIXMAANH	15	1e−1	167	5.656e−1	54	9.557e−1	394	2.593e−2
DIXMAANH	15	1e−3	280	7.738e−3	395	4.275e−3	705	3.527e−5
DIXMAANH	15	1e−5	365	3.603e−5	419	6.578e−5	825	7.652e−8
DIXMAANH	15	1e−7	400	1.836e−7	767	2.555e−7	830	9.052e−10
DIXMAANH	90	1e−1	1598	2.375	963	1.997	3163	8.808e−2
DIXMAANH	90	1e−3	2287	1.070e−1	3112	9.064e−2	8780	3.029e−3
DIXMAANH	90	1e−5	3252	2.565e−4	5492	8.586e−4	6194	4.423e−6
DIXMAANH	90	1e−7	5217	3.169e−6	4430	6.061e−6	8780	4.295e−9
DIXMAANH	300	1e−1	19419	5.492	1211	1.205e1	6759	3.222e−1
DIXMAANH	300	1e−3	33582	2.678e−1	15009	1.093e−1	17702	4.861e−3
DIXMAANH	300	1e−5	25154	4.098e−3	15460	4.812e−3	40314	1.688e−5
DIXMAANH	300	1e−7	25753	1.920e−5	22065	2.289e−5	41288	8.904e−8
DIXMAANI	15	1e−1	113	5.926e−1	1218	1.032	516	3.909e−2
DIXMAANI	15	1e−3	358	6.615e−3	504	1.797e−2	1053	2.345e−3
DIXMAANI	15	1e−5	467	6.109e−4	1343	3.261e−4	1476	1.374e−6
DIXMAANI	15	1e−7	636	6.946e−6	792	1.308e−6	1611	4.603e−9
DIXMAANI	90	1e−1	951	2.135	3000	4.382	7696	2.633e−1
DIXMAANI	90	1e−3	1824	4.215e−2	2824	5.241e−2	7466	5.757e−3
DIXMAANI	90	1e−5	3278	7.114e−3	5618	4.747e−3	18792	1.816e−4
DIXMAANI	90	1e−7	10692	2.134e−4	12783	1.281e−4	36022	5.632e−7
DIXMAANI	300	1e−1	7480	5.731	12317	1.749e1	21554	9.401e−1
DIXMAANI	300	1e−3	12194	1.431e−1	10792	1.467e−1	43977	1.445e−2
DIXMAANI	300	1e−5	25477	9.917e−3	34341	6.574e−3	70880	3.511e−4
DIXMAANI	300	1e−7	36578	1.928e−4	74070	2.406e−4	150151*	1.779e−5
DIXMAANJ	15	1e−1	156	1.786e−1	52	6.219e−1	1124	3.543e−2
DIXMAANJ	15	1e−3	309	2.567e−2	584	1.557e−2	1536	7.979e−4
DIXMAANJ	15	1e−5	672	7.838e−5	835	2.748e−4	1124	7.681e−7
DIXMAANJ	15	1e−7	685	4.621e−7	789	3.246e−5	2031	1.804e−8
DIXMAANJ	90	1e−1	1141	1.187	277	3.668	1880	7.149e−2
DIXMAANJ	90	1e−3	2186	4.912e−2	2853	5.441e−2	7696	2.858e−3
DIXMAANJ	90	1e−5	4146	3.591e−3	3791	3.090e−3	11399	4.242e−5
DIXMAANJ	90	1e−7	13470	6.305e−4	7251	3.692e−5	29130	2.883e−7
DIXMAANJ	300	1e−1	4325	2.391	904	1.236e1	11680	1.834e−1
DIXMAANJ	300	1e−3	23360	1.285e−1	8042	1.123e−1	36345	8.119e−3
DIXMAANJ	300	1e−5	19573	3.531e−3	18352	5.437e−3	41369	5.491e−5
DIXMAANJ	300	1e−7	31059	1.801e−4	33637	5.060e−5	84610	2.777e−6

Table 13: Noisy Unconstrained `CUTEst` Problems Tested. n is the number of variables. σ_f is the standard deviation of the noise.

			NEWUOA		FD L-BFGS		CD L-BFGS	
Problem	n	σ_f	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$
DIXMAANK	15	1e−1	123	2.036e−1	70	3.310e−1	270	5.345e−2
DIXMAANK	15	1e−3	226	4.148e−2	1229	1.660e−2	1024	8.734e−4
DIXMAANK	15	1e−5	533	4.054e−4	759	7.894e−5	1730	3.452e−7
DIXMAANK	15	1e−7	710	1.111e−6	724	3.219e−7	1595	4.578e−9
DIXMAANK	90	1e−1	962	2.115	961	9.536e−1	1880	3.049e−1
DIXMAANK	90	1e−3	1921	4.371e−2	2718	9.449e−2	10211	3.178e−3
DIXMAANK	90	1e−5	3646	8.827e−4	7155	1.234e−3	8157	3.217e−5
DIXMAANK	90	1e−7	6961	5.780e−4	5158	2.208e−5	14626	1.152e−6
DIXMAANK	300	1e−1	4131	3.657	908	6.765	12777	4.528e−1
DIXMAANK	300	1e−3	30156	1.396e−1	10483	3.126e−1	21925	3.813e−3
DIXMAANK	300	1e−5	17158	7.029e−3	16876	3.556e−3	26675	6.708e−5
DIXMAANK	300	1e−7	52657	1.527e−4	15408	7.813e−5	82360	6.627e−7
DIXMAANL	15	1e−1	152	1.451	88	5.246e−1	1246	5.972e−2
DIXMAANL	15	1e−3	370	3.861e−2	866	1.485e−2	1694	5.900e−4
DIXMAANL	15	1e−5	782	6.330e−5	1050	1.029e−4	1476	1.103e−6
DIXMAANL	15	1e−7	873	1.152e−6	638	1.039e−6	1730	2.064e−8
DIXMAANL	90	1e−1	1230	1.224	1363	1.218	2817	1.846e−1
DIXMAANL	90	1e−3	2191	4.680e−2	2539	1.429e−1	5193	2.433e−3
DIXMAANL	90	1e−5	3048	1.131e−3	3152	2.455e−3	8780	3.050e−5
DIXMAANL	90	1e−7	7705	2.888e−4	5990	1.848e−5	32927	1.567e−7
DIXMAANL	300	1e−1	7660	3.606	1211	6.900	11674	1.739e−1
DIXMAANL	300	1e−3	33194	1.813e−1	11079	2.767e−1	19562	4.733e−3
DIXMAANL	300	1e−5	149998	3.705e−2	10571	5.175e−3	26767	4.075e−5
DIXMAANL	300	1e−7	61830	7.296e−4	24171	5.364e−5	57101	1.891e−6
DQRTIC	10	1e−1	139	1.786e−1	927	1.449e−1	762	2.816e−4
DQRTIC	10	1e−3	152	4.676e−3	304	6.109e−3	749	3.770e−5
DQRTIC	10	1e−5	429	8.020e−5	943	3.905e−5	675	8.733e−9
DQRTIC	10	1e−7	405	3.782e−8	676	1.675e−6	675	1.032e−8
DQRTIC	50	1e−1	1672	7.132e−1	1197	8.681e−1	2857	7.958e−2
DQRTIC	50	1e−3	1719	7.324e−3	2008	7.825e−3	2651	3.038e−4
DQRTIC	50	1e−5	2369	2.603e−5	2629	1.445e−4	3341	1.758e−6
DQRTIC	50	1e−7	2435	1.473e−6	2493	1.915e−6	3341	2.992e−8
DQRTIC	100	1e−1	4920	3.885e−1	4766	6.422	7146	4.248e−2
DQRTIC	100	1e−3	5612	9.569e−3	4517	2.474e−3	5960	1.807e−4
DQRTIC	100	1e−5	6494	5.767e−5	6534	7.941e−4	7146	2.060e−6
DQRTIC	100	1e−7	6913	8.633e−7	1119	1.906e7	7344	2.041e−8
EDENSCH	36	1e−1	873	9.721e−1	1071	1.223	2104	1.290e−1
EDENSCH	36	1e−3	1175	1.187e−2	2031	1.116e−2	1772	3.795e−4
EDENSCH	36	1e−5	1166	1.001e−4	1324	5.176e−4	1772	8.173e−7
EDENSCH	36	1e−7	1331	1.083e−6	1300	3.572e−6	1824	2.417e−9
EIGENALS	6	1e−1	5**	0.000	96	5.364e−1	509	2.108e−1
EIGENALS	6	1e−3	5**	0.000	82	1.041e−2	252	5.299e−4
EIGENALS	6	1e−5	5**	0.000	250	4.298e−4	152	1.284e−7
EIGENALS	6	1e−7	5**	0.000	294	5.331e−6	395	1.289e−10
EIGENALS	110	1e−1	2393	9.021	3176	1.692e1	8820	1.404e−1
EIGENALS	110	1e−3	6471	8.081e−1	3365	2.275e−1	5137	1.253e−2
EIGENALS	110	1e−5	36272	2.114e−2	6187	1.407e−2	41789	2.313e−4
EIGENALS	110	1e−7	54998	2.423e−4	32572	1.418e−4	55103*	2.471e−5

Table 14: Noisy Unconstrained `CUTEst` Problems Tested. n is the number of variables. σ_f is the standard deviation of the noise.

			NEWUOA		FD L-BFGS		CD L-BFGS	
Problem	n	σ_f	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$
EIGENBLS‡	6	1e−1	51	1.421e−1	32	1.096	252	3.986e−3
EIGENBLS‡	6	1e−3	109	3.843e−2	240	1.223e−3	376	4.336e−5
EIGENBLS‡	6	1e−5	217	−1.840e−1	247	2.792e−5	299	1.340e−7
EIGENBLS‡	6	1e−7	307	−1.849e−1	190	9.576e−8	256	5.404e−10
EIGENBLS	110	1e−1	705	3.389	1761	1.122e1	4023	1.707
EIGENBLS	110	1e−3	5295	1.452	2754	1.596	7292	1.552
EIGENBLS	110	1e−5	18185	2.700e−2	21793	2.398e−2	55096*	5.814e−3
EIGENBLS	110	1e−7	30763	5.299e−4	46722	9.959e−4	55088*	2.591e−2
EIGENCLS	30	1e−1	400	1.610	1107	1.185e1	1812	4.235e−1
EIGENCLS	30	1e−3	686	3.905e−1	1546	2.774e−2	3424	1.360e−3
EIGENCLS	30	1e−5	1590	1.176e−3	1935	1.016e−3	4427	1.116e−5
EIGENCLS	30	1e−7	2212	6.465e−6	3159	1.266e−5	6325	3.770e−8
ENGVAL1	2	1e−1	22	3.602e−2	24	5.487e−1	146	1.987e−2
ENGVAL1	2	1e−3	30	9.036e−5	203	4.722e−3	146	2.480e−5
ENGVAL1	2	1e−5	33	5.987e−8	54	2.397e−6	91	2.265e−8
ENGVAL1	2	1e−7	36	3.361e−9	56	3.129e−8	91	1.713e−11
ENGVAL1	50	1e−1	1298	1.868	2612	6.147	1160	2.136e−1
ENGVAL1	50	1e−3	1125	2.268e−2	634	1.857e−2	3720	2.678e−4
ENGVAL1	50	1e−5	2643	1.160e−4	894	5.713e−5	1460	2.359e−6
ENGVAL1	50	1e−7	2802	9.129e−7	1354	1.628e−6	2651	2.889e−9
ENGVAL1	100	1e−1	2362	1.585	3555	1.187e1	1638	3.397e−1
ENGVAL1	100	1e−3	3400	1.462e−2	2805	4.521e−2	4885	8.850e−4
ENGVAL1	100	1e−5	3501	1.729e−4	1334	5.313e−4	5960	2.445e−6
ENGVAL1	100	1e−7	3166	1.293e−6	4913	3.878e−6	4095	6.088e−9
EXPFIT	2	1e−1	17	3.207	54	4.954	126	4.794e−2
EXPFIT	2	1e−3	33	3.946e−3	58	4.299e−3	67	4.256e−5
EXPFIT	2	1e−5	38	3.463e−7	107	2.064e−6	260	7.468e−10
EXPFIT	2	1e−7	49	8.778e−10	68	2.677e−8	91	5.019e−11
FLETCBV3‡	10	1e−1	21**	3.228e−2	32	3.228e−2	21	3.228e−2
FLETCBV3‡	10	1e−3	21**	3.228e−2	29	3.228e−2	21	3.228e−2
FLETCBV3‡	10	1e−5	22	3.228e−2	27	3.228e−2	21	3.228e−2
FLETCBV3‡	10	1e−7	965	8.774e−4	27	3.228e−2	403	3.228e−2
FLETCBV3‡	100	1e−1	202	1.785e5	2486	1.785e5	1952	1.785e5
FLETCBV3‡	100	1e−3	202	1.785e5	447	1.785e5	4029	1.785e5
FLETCBV3‡	100	1e−5	50000*	1.349e5	11892	1.785e5	49799	2.028e2
FLETCBV3‡	100	1e−7	49999	5.761e4	222	1.785e5	50010*	1.015e1
FLETCHBV‡	10	1e−1	863	2.240e4	639	2.653e3	1087	1.218e4
FLETCHBV‡	10	1e−3	1289	1.540e5	1118	8.389e3	1356	−4.764e3
FLETCHBV‡	10	1e−5	790	2.228e5	708	−4.605e3	1445	−7.519e3
FLETCHBV‡	10	1e−7	977	8.461e4	759	7.233e2	952	4.236e2
FLETCHBV‡	100	1e−1	50000*	1.581e13	40962	−7.187e9	50076*	−6.387e9
FLETCHBV‡	100	1e−3	50000*	1.731e13	50015*	−9.179e9	50085*	−3.093e9
FLETCHBV‡	100	1e−5	50000*	1.704e13	33119	−8.684e8	50068*	−5.710e9
FLETCHBV‡	100	1e−7	50000*	1.391e13	8369	7.425e10	50080*	−5.308e9

Table 15: Noisy Unconstrained `CUTEst` Problems Tested. n is the number of variables. σ_f is the standard deviation of the noise.

			NEWUOA		FD L-BFGS		CD L-BFGS	
Problem	n	σ_f	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$
FREUROTH	2	1e−1	38	6.304e−2	35	5.015e1	368	3.877e−4
FREUROTH	2	1e−3	35	5.507e−1	347	1.660e−3	141	1.496e−5
FREUROTH	2	1e−5	70	8.659e−7	83	9.381e−5	387	3.302e−8
FREUROTH	2	1e−7	73	4.594e−10	138	1.112e−6	261	4.996e−10
FREUROTH	10	1e−1	125	4.555e1	137	5.548e1	1466	8.708e−3
FREUROTH	10	1e−3	371	3.964e−3	467	1.341e−1	562	4.686e−5
FREUROTH	10	1e−5	433	3.004e−4	295	1.857e−3	842	5.755e−8
FREUROTH	10	1e−7	435	1.512e−6	602	1.348e−5	630	−3.206e−10
FREUROTH	50	1e−1	1629	5.744e1	2466	5.503e1	2651	6.734e−2
FREUROTH	50	1e−3	6822	6.137	3983	3.723e−2	2857	5.299e−5
FREUROTH	50	1e−5	7662	1.340e−3	5024	3.387e−4	2487	2.101e−7
FREUROTH	50	1e−7	6619	9.762e−6	1257	1.453e−6	3063	5.975e−10
FREUROTH	100	1e−1	1868	5.751e1	920	6.619e1	8609	6.268e−2
FREUROTH	100	1e−3	8308	9.577	3192	5.430e−2	6050	1.752e−4
FREUROTH	100	1e−5	5029	1.824e−3	4095	2.290e−4	5769	4.168e−7
FREUROTH	100	1e−7	5293	7.505e−6	6187	1.098e−5	5907	−5.191e−9
GENROSE	5	1e−1	35	3.849	68	3.246	180	2.711
GENROSE	5	1e−3	162	4.182e−2	234	2.566	446	9.023e−4
GENROSE	5	1e−5	221	8.543e−5	260	2.548e−3	413	2.278e−7
GENROSE	5	1e−7	261	9.023e−7	317	4.476e−5	454	1.113e−9
GENROSE	10	1e−1	84	9.259	221	8.902	805	8.892
GENROSE	10	1e−3	585	4.321e−3	446	8.816	1564	1.860e−4
GENROSE	10	1e−5	654	3.114e−5	1016	7.485e−3	1935	1.029e−6
GENROSE	10	1e−7	726	7.303e−7	964	8.881e−5	1904	5.181e−10
GENROSE	100	1e−1	1868	1.189e2	3231	1.335e2	7344	1.119e2
GENROSE	100	1e−3	15359	1.081e2	2247	1.108e2	50024*	6.559e1
GENROSE	100	1e−5	18075	6.885e−3	29786	5.081e−3	50009*	2.770e−1
GENROSE	100	1e−7	18030	1.719e−6	29847	4.748e−5	50005*	2.194e−2
GULF	3	1e−1	26	7.047	41	6.637	389	6.622
GULF	3	1e−3	38	7.032	38	6.622	229	4.534e−3
GULF	3	1e−5	115	6.882e−2	267	3.649e−3	370	4.462e−3
GULF	3	1e−7	333	6.930e−3	217	4.381e−3	688	2.460e−7
HAIRY	2	1e−1	49	3.471e2	71	2.721e2	387	1.950e−4
HAIRY	2	1e−3	351	2.426e−4	186	8.522e−4	413	2.326e−8
HAIRY	2	1e−5	226	1.256e−8	175	9.401e−7	516	5.074e−11
HAIRY	2	1e−7	252	2.339e−8	87	1.508e−6	210	4.974e−14
HELIX	3	1e−1	26	9.616e−2	188	7.615	374	2.785e−1
HELIX	3	1e−3	38	1.865e−2	314	4.147	23	8.522e2
HELIX	3	1e−5	72	1.822e−6	453	3.021e−6	23	8.521e2
HELIX	3	1e−7	84	4.567e−7	167	1.171e−4	23	8.521e2
JENSMP‡	2	1e−1	24	−1.786e3	18	0.000	21	0.000
JENSMP‡	2	1e−3	78	−1.896e3	18	0.000	21	0.000
JENSMP‡	2	1e−5	87	−1.896e3	18	0.000	21	0.000
JENSMP‡	2	1e−7	67	−1.896e3	18	0.000	21	0.000
KOWOSB	4	1e−1	1**	5.006e−3	72	1.383e−3	9	5.006e−3
KOWOSB	4	1e−3	20	2.240e−3	261	3.305e−3	104	2.435e−4
KOWOSB	4	1e−5	44	1.249e−4	112	1.767e−4	220	6.118e−5
KOWOSB	4	1e−7	111	2.694e−5	1309	6.915e−7	314	5.638e−8

Table 16: Noisy Unconstrained `CUTEst` Problems Tested. n is the number of variables. σ_f is the standard deviation of the noise.

			NEWUOA		FD L-BFGS		CD L-BFGS	
Problem	n	σ_f	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$
MEXHAT	2	1e−1	30	1.166e−2	98	4.477e−1	146	4.231e−1
MEXHAT	2	1e−3	43	1.102e−2	80	8.450e−1	709	4.227e−1
MEXHAT	2	1e−5	45	1.110e−2	203	4.173e−1	232	1.568e−3
MEXHAT	2	1e−7	44	1.036e−2	210	4.226e−1	660	1.383e−4
MOREBV	10	1e−1	1**	1.599e−2	19	1.599e−2	21	1.599e−2
MOREBV	10	1e−3	81	1.262e−2	131	1.451e−2	937	1.506e−3
MOREBV	10	1e−5	184	2.345e−3	433	1.847e−3	796	1.441e−6
MOREBV	10	1e−7	386	1.283e−7	988	7.402e−5	1003	5.675e−9
MOREBV	50	1e−1	1**	1.321e−3	93	1.321e−3	101	1.321e−3
MOREBV	50	1e−3	105	1.122e−3	101	1.321e−3	828	4.204e−4
MOREBV	50	1e−5	746	3.981e−4	997	7.841e−4	2064	2.290e−5
MOREBV	50	1e−7	1322	3.021e−5	1283	4.621e−5	12825	7.628e−6
MOREBV	100	1e−1	1**	3.633e−4	190	3.633e−4	201	3.633e−4
MOREBV	100	1e−3	1**	3.633e−4	201	3.633e−4	2669	1.495e−4
MOREBV	100	1e−5	320	2.538e−4	309	3.571e−4	3457	6.772e−6
MOREBV	100	1e−7	3087	1.417e−5	2809	1.540e−5	14045	1.546e−6
NCB20B	21	1e−1	1**	3.000e−3	22	3.000e−3	232	3.177e−4
NCB20B	21	1e−3	113	1.441e−5	74	2.964e−4	421	6.287e−6
NCB20B	21	1e−5	260	1.402e−4	1332	5.741e−6	686	6.098e−9
NCB20B	21	1e−7	708	1.224e−5	260	4.816e−8	650	−5.574e−11
NCB20B	22	1e−1	121	5.384e−3	23	6.000e−3	388	1.036e−3
NCB20B	22	1e−3	121	1.817e−3	174	8.594e−4	935	8.921e−4
NCB20B	22	1e−5	265	2.971e−4	298	1.075e−5	578	3.829e−6
NCB20B	22	1e−7	715	2.307e−5	345	1.163e−7	908	2.372e−9
NCB20B	50	1e−1	212	9.199e−2	121	1.099e−1	938	1.883e−2
NCB20B	50	1e−3	734	7.836e−3	1513	1.296e−3	1866	3.565e−4
NCB20B	50	1e−5	1757	4.549e−4	838	3.965e−4	3623	9.630e−5
NCB20B	50	1e−7	4316	6.696e−5	2253	8.619e−5	23370	2.289e−6
NCB20B	100	1e−1	481	3.871e−1	1565	3.458e−1	2938	3.874e−2
NCB20B	100	1e−3	1748	1.447e−2	2557	2.895e−2	5703	8.588e−4
NCB20B	100	1e−5	5976	7.143e−4	5154	3.922e−4	10509	8.218e−5
NCB20B	100	1e−7	15651	4.301e−5	6944	6.476e−5	32343	1.546e−5
NCB20B	180	1e−1	1664	4.700e−1	1648	1.184	5833	1.772e−1
NCB20B	180	1e−3	6604	4.261e−2	2007	5.420e−2	12753	1.807e−3
NCB20B	180	1e−5	12066	1.067e−3	10462	3.299e−4	22900	2.856e−5
NCB20B	180	1e−7	26507	2.789e−5	11118	2.804e−5	28345	2.440e−5
NONDIA	10	1e−1	84	4.198	407	6.547	124	6.487
NONDIA	10	1e−3	144	1.406e−3	79	6.481	579	1.101e−4
NONDIA	10	1e−5	168	3.507e−5	93	6.480	349	5.342e−7
NONDIA	10	1e−7	195	2.695e−7	185	1.575e−7	322	4.429e−10
NONDIA	20	1e−1	154	2.666	814	4.149	885	4.485
NONDIA	20	1e−3	356	2.292e−3	140	4.435	744	5.560e−4
NONDIA	20	1e−5	403	3.163e−5	382	1.709e−3	996	4.738e−7
NONDIA	20	1e−7	468	1.399e−6	383	4.588e−7	1933	3.743e−9
NONDIA	30	1e−1	199	1.497	169	3.286	517	3.271
NONDIA	30	1e−3	566	6.818e−3	780	1.606e−2	1319	1.828e−3
NONDIA	30	1e−5	589	1.475e−4	689	8.898e−4	1082	5.285e−6
NONDIA	30	1e−7	634	3.771e−7	1319	6.491e−7	1551	4.326e−9

Table 17: Noisy Unconstrained `CUTEst` Problems Tested. n is the number of variables. σ_f is the standard deviation of the noise.

			NEWUOA		FD L-BFGS		CD L-BFGS	
Problem	n	σ_f	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$
NONDIA	50	1e−1	317	8.665e−1	269	1.505	423	1.520
NONDIA	50	1e−3	635	8.814e−3	375	1.498	1659	2.467e−3
NONDIA	50	1e−5	851	7.108e−5	3599	4.640e−4	1659	6.908e−6
NONDIA	50	1e−7	1132	1.517e−6	1731	1.971e−5	1556	1.127e−8
NONDIA	90	1e−1	573	3.893e−1	1087	6.154e−1	744	5.843e−1
NONDIA	90	1e−3	1374	3.301e−1	563	5.766e−1	2052	9.232e−3
NONDIA	90	1e−5	1720	9.430e−5	564	5.781e−1	3319	2.610e−5
NONDIA	90	1e−7	1788	2.403e−6	1306	4.184e−6	4057	5.888e−8
NONDIA	100	1e−1	419	3.342e−1	418	4.850e−1	824	4.930e−1
NONDIA	100	1e−3	1610	2.086e−1	927	4.803e−1	4544	1.815e−2
NONDIA	100	1e−5	1907	1.474e−4	934	4.658e−1	4975	4.317e−5
NONDIA	100	1e−7	2235	2.091e−6	2668	5.756e−5	4072	1.010e−7
NONDQUAR	100	1e−1	828	5.744e−1	1433	7.564e−1	3532	1.006e−1
NONDQUAR	100	1e−3	1842	1.889e−2	5568	2.000e−2	7272	5.244e−3
NONDQUAR	100	1e−5	4661	1.475e−3	8339	1.946e−3	20194	3.394e−4
NONDQUAR	100	1e−7	26783	1.700e−4	21822	1.549e−4	50037*	3.497e−5
OSBORNEA‡	5	1e−1	52	1.127e−2	8	8.790e−1	170	1.485e−1
OSBORNEA‡	5	1e−3	64	5.240e−2	9	8.790e−1	496	7.589e−4
OSBORNEA‡	5	1e−5	135	3.040e−3	260	1.902e−3	1638	2.432e−5
OSBORNEA‡	5	1e−7	504	4.404e−5	677	2.252e−5	1033	2.247e−5
OSBORNEB	11	1e−1	89	5.190e−1	308	6.677e−1	1031	2.972e−1
OSBORNEB	11	1e−3	172	3.075e−1	829	3.219e−1	1238	7.480e−2
OSBORNEB	11	1e−5	1052	7.968e−3	1249	2.882e−3	2334	7.854e−5
OSBORNEB	11	1e−7	1708	8.191e−6	1555	6.681e−6	2395	9.143e−9
PENALTY1	4	1e−1	49	1.917e−2	40	8.686e−3	100	4.980e−3
PENALTY1	4	1e−3	87	2.835e−5	165	3.661e−5	95	4.498e−5
PENALTY1	4	1e−5	117	3.573e−5	55	1.741e−4	106	3.829e−5
PENALTY1	4	1e−7	113	2.254e−5	97	3.820e−5	140	3.826e−5
PENALTY1	10	1e−1	358	4.733e−1	1167	5.627e−2	284	1.327e−3
PENALTY1	10	1e−3	178	3.567e−5	347	2.207e−4	353	9.096e−5
PENALTY1	10	1e−5	194	2.463e−5	335	6.078e−5	595	6.005e−5
PENALTY1	10	1e−7	245	2.985e−5	352	5.882e−5	477	5.965e−5
PENALTY1	50	1e−1	2527	5.606e−1	2274	1.423	2076	8.180e−3
PENALTY1	50	1e−3	2455	6.140e−5	2697	1.201e−4	2281	2.999e−4
PENALTY1	50	1e−5	2298	5.576e−5	1786	9.926e−5	2384	1.322e−4
PENALTY1	50	1e−7	3919	8.208e−6	1523	8.616e−5	16116	7.338e−6
PENALTY1	100	1e−1	5649	8.758e−1	6442	6.236	4687	8.428e−3
PENALTY1	100	1e−3	6701	1.019e−4	3923	1.489e−4	5093	1.859e−4
PENALTY1	100	1e−5	5940	9.287e−5	4516	1.904e−4	5907	1.878e−4
PENALTY1	100	1e−7	10954	1.256e−5	6607	1.677e−4	28399	5.191e−6
PFIT1LS‡	3	1e−1	35	4.034	67	9.281	389	5.246
PFIT1LS‡	3	1e−3	111	9.255e−1	1415	7.002	447	2.288e−4
PFIT1LS‡	3	1e−5	325	1.154e−2	342	2.235e−6	502	2.581e−6
PFIT1LS‡	3	1e−7	482	2.955e−4	258	4.493e−5	453	2.489e−6
PFIT2LS‡	3	1e−1	113	5.174e1	81	9.874e1	620	2.712e−1
PFIT2LS‡	3	1e−3	410	2.136	123	1.245e2	537	6.269e−3
PFIT2LS‡	3	1e−5	618	5.570e−2	430	4.757e−3	537	1.772e−3
PFIT2LS‡	3	1e−7	706	1.251e−2	340	2.704e−3	870	1.648e−3

Table 18: Noisy Unconstrained `CUTEst` Problems Tested. n is the number of variables. σ_f is the standard deviation of the noise.

			NEWUOA		FD L-BFGS		CD L-BFGS	
Problem	n	σ_f	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$
PFIT3LS‡	3	1e−1	162	1.868e2	390	7.944e1	817	3.682e−1
PFIT3LS‡	3	1e−3	546	7.561	640	9.159e−1	612	3.276e−2
PFIT3LS‡	3	1e−5	1012	9.384e−2	615	4.587e−2	702	3.154e−2
PFIT3LS‡	3	1e−7	1012	8.229e−2	444	4.374e−2	1504*	2.782e−2
PFIT4LS‡	3	1e−1	234	2.464e2	78	2.144e3	723	1.842e−1
PFIT4LS‡	3	1e−3	612	2.736e1	758	6.570	715	1.234e−1
PFIT4LS‡	3	1e−5	1181	8.871e−1	471	1.677e−1	1506*	1.433e−1
PFIT4LS‡	3	1e−7	1374	2.619e−1	597	1.249e−1	1502*	1.086e−1
QUARTC	25	1e−1	599	4.172e−1	1716	2.004	1203	1.376e−2
QUARTC	25	1e−3	759	7.392e−3	767	3.811e−2	1496	8.968e−5
QUARTC	25	1e−5	828	6.201e−6	1245	8.620e−5	1750	3.508e−7
QUARTC	25	1e−7	1040	4.136e−7	1584	1.148e−6	2807	1.313e−8
QUARTC	100	1e−1	4920	3.885e−1	4766	6.422	7146	4.248e−2
QUARTC	100	1e−3	5612	9.569e−3	4517	2.474e−3	5960	1.807e−4
QUARTC	100	1e−5	6494	5.767e−5	6534	7.941e−4	7146	2.060e−6
QUARTC	100	1e−7	6913	8.633e−7	1119	1.906e7	7344	2.041e−8
SINEVAL	2	1e−1	12	5.231	4	5.552	108	4.632
SINEVAL	2	1e−3	152	9.339e−1	194	3.587	700	2.082e−4
SINEVAL	2	1e−5	226	1.682e−5	450	1.856e−1	488	1.360e−7
SINEVAL	2	1e−7	295	1.841e−8	425	1.955e−5	563	9.899e−13
SINQUAD	5	1e−1	36	1.503	333	5.219	107	1.121e−2
SINQUAD	5	1e−3	98	8.682e−3	79	6.186e−3	121	3.326e−5
SINQUAD	5	1e−5	140	1.602e−5	77	3.260e−5	121	9.620e−8
SINQUAD	5	1e−7	170	1.177e−6	135	5.826e−9	205	−3.938e−10
SINQUAD	50	1e−1	1724	1.740	742	2.222e1	3929	1.376e−1
SINQUAD	50	1e−3	3724	3.119e−2	1101	9.615e−3	3616	2.544e−4
SINQUAD	50	1e−5	4234	3.236e−4	849	2.589e−4	2857	6.467e−7
SINQUAD	50	1e−7	6464	1.670e−6	1261	1.644e−6	2083	5.443e−9
SINQUAD	100	1e−1	4978	4.096	1247	1.048e1	3062	1.026e−1
SINQUAD	100	1e−3	7193	2.876e−2	2470	6.295e−2	3265	2.132e−4
SINQUAD	100	1e−5	10426	3.206e−4	1948	9.257e−4	3468	5.694e−7
SINQUAD	100	1e−7	15008	2.263e−6	2152	3.046e−6	4079	3.929e−9
SISSER	2	1e−1	4**	3.000e−4	20	2.326e−2	65	2.424e−4
SISSER	2	1e−3	15	1.764e−4	139	2.826e−5	204	1.415e−4
SISSER	2	1e−5	20	2.270e−5	44	3.731e−5	85	2.188e−9
SISSER	2	1e−7	27	1.467e−8	71	4.270e−7	116	6.695e−12
SPARSQUR	10	1e−1	42	2.779e−1	104	4.304e−2	403	5.561e−3
SPARSQUR	10	1e−3	85	2.675e−3	212	6.281e−3	384	6.274e−5
SPARSQUR	10	1e−5	184	4.915e−5	740	2.770e−5	384	3.317e−7
SPARSQUR	10	1e−7	241	1.377e−6	398	1.386e−7	579	1.307e−8
SPARSQUR	50	1e−1	1070	7.980e−1	1303	6.397e−1	1316	1.918e−2
SPARSQUR	50	1e−3	1034	4.918e−3	1078	1.952e−2	3577	9.425e−5
SPARSQUR	50	1e−5	1472	2.315e−5	2755	4.135e−5	2651	5.338e−7
SPARSQUR	50	1e−7	1847	3.804e−7	1706	5.903e−6	2651	6.858e−9
SPARSQUR	100	1e−1	2912	5.988e−1	3387	1.953	5960	1.100e−2
SPARSQUR	100	1e−3	2484	4.672e−3	2435	5.029e−2	3677	5.412e−5
SPARSQUR	100	1e−5	2664	7.151e−5	5794	1.737e−4	4292	6.552e−6
SPARSQUR	100	1e−7	3757	3.002e−7	4591	2.610e−6	5960	1.082e−8

Table 19: Noisy Unconstrained `CUTEst` Problems Tested. n is the number of variables. σ_f is the standard deviation of the noise.

			NEWUOA		FD L-BFGS		CD L-BFGS	
Problem	n	σ_f	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$
TOINTGSS	10	1e−1	65	4.850e−1	759	2.072	101	1.831e−2
TOINTGSS	10	1e−3	164	1.448e−2	301	3.304e−2	165	8.167e−6
TOINTGSS	10	1e−5	211	2.845e−5	124	3.624e−4	200	2.744e−8
TOINTGSS	10	1e−7	277	2.113e−7	124	3.539e−6	200	5.802e−11
TOINTGSS	50	1e−1	388	1.033	563	1.183e1	724	6.155e−2
TOINTGSS	50	1e−3	878	1.276e−2	415	4.651e−2	2132	1.554e−5
TOINTGSS	50	1e−5	923	1.748e−4	693	1.140e−3	724	5.405e−8
TOINTGSS	50	1e−7	1574	1.030e−6	693	1.136e−5	724	−2.877e−9
TOINTGSS	100	1e−1	1638	1.693	590	7.090	2660	6.483e−2
TOINTGSS	100	1e−3	1521	2.165e−2	509	7.545e−2	1646	3.228e−5
TOINTGSS	100	1e−5	2692	2.385e−4	508	1.898e−3	2526	2.620e−8
TOINTGSS	100	1e−7	2312	3.442e−6	508	1.893e−5	2526	4.046e−9
TQUARTIC	5	1e−1	22	4.614e−1	24	6.142e−1	40	4.540e−1
TQUARTIC	5	1e−3	63	5.640e−3	242	9.902e−2	234	5.290e−4
TQUARTIC	5	1e−5	107	1.155e−5	202	8.160e−4	818	1.004e−6
TQUARTIC	5	1e−7	137	7.045e−8	95	1.098e−6	163	9.192e−9
TQUARTIC	10	1e−1	44	6.151e−1	274	7.097e−1	92	5.464e−1
TQUARTIC	10	1e−3	174	4.406e−2	543	2.485e−1	344	5.223e−3
TQUARTIC	10	1e−5	277	2.548e−5	466	4.217e−2	722	7.389e−6
TQUARTIC	10	1e−7	388	8.693e−7	805	6.160e−5	535	4.256e−8
TQUARTIC	50	1e−1	191	6.732e−1	170	7.480e−1	310	7.002e−1
TQUARTIC	50	1e−3	1487	3.648e−1	1155	5.611e−1	1534	6.796e−1
TQUARTIC	50	1e−5	3112	6.435e−3	1441	2.121e−2	2117	1.390e−4
TQUARTIC	50	1e−7	5704	8.027e−5	3736	3.333e−4	1785	3.649e−7
TQUARTIC	100	1e−1	226	7.236e−1	236	7.828e−1	610	7.423e−1
TQUARTIC	100	1e−3	1085	7.089e−1	790	7.232e−1	3629	7.219e−1
TQUARTIC	100	1e−5	26574	2.458e−2	2675	1.765e−1	4298	5.460e−4
TQUARTIC	100	1e−7	31937	2.808e−4	4860	4.761e−4	5838	2.894e−7
TRIDIA	10	1e−1	128	1.555	124	1.392	594	3.847e−4
TRIDIA	10	1e−3	174	1.649e−3	305	2.065e−2	495	1.429e−6
TRIDIA	10	1e−5	176	2.353e−5	242	3.449e−4	698	4.786e−9
TRIDIA	10	1e−7	226	1.205e−7	279	3.036e−6	848	2.043e−12
TRIDIA	20	1e−1	301	7.391e−1	812	1.072	2309	4.788e−4
TRIDIA	20	1e−3	312	9.505e−3	443	1.352e−1	1432	1.038e−6
TRIDIA	20	1e−5	396	4.890e−5	821	4.616e−4	2029	3.474e−9
TRIDIA	20	1e−7	500	5.289e−7	864	3.588e−6	1992	7.669e−12
TRIDIA	30	1e−1	454	8.863e−1	3249	7.566	2694	2.364e−3
TRIDIA	30	1e−3	599	7.428e−3	2486	6.494e−2	2403	4.586e−5
TRIDIA	30	1e−5	753	1.083e−4	1254	1.150e−3	3111	7.820e−7
TRIDIA	30	1e−7	849	7.455e−7	2292	5.997e−6	4598	6.846e−11
TRIDIA	50	1e−1	670	1.625	1604	8.841	3723	1.190e−1
TRIDIA	50	1e−3	1124	1.616e−2	1824	2.452e−1	6195	7.991e−6
TRIDIA	50	1e−5	1056	4.101e−4	3602	1.237e−3	5985	2.190e−6
TRIDIA	50	1e−7	1784	1.022e−6	3450	2.572e−5	7640	1.634e−8
TRIDIA	100	1e−1	2183	5.214	4778	2.202e1	9774	1.524e−1
TRIDIA	100	1e−3	4301	3.587e−2	3474	6.746e−1	15035	9.421e−5
TRIDIA	100	1e−5	4682	4.705e−4	10097	2.340e−3	20327	4.890e−7
TRIDIA	100	1e−7	3650	4.093e−6	11716	3.468e−5	22968	8.918e−9

Table 20: Noisy Unconstrained `CUTEst` Problems Tested. n is the number of variables. σ_f is the standard deviation of the noise.

			NEWUOA		FD L-BFGS		CD L-BFGS	
Problem	n	σ_f	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$	#feval	$\phi(x)-\phi^*$
WATSON	12	1e−1	91	1.602e−1	260	2.747e−1	407	9.362e−2
WATSON	12	1e−3	170	7.112e−2	752	7.265e−3	737	8.831e−3
WATSON	12	1e−5	708	6.531e−4	1487	2.403e−3	1470	1.808e−4
WATSON	12	1e−7	1664	2.355e−4	1077	1.868e−4	1559	1.367e−5
WATSON	31	1e−1	283	1.444	567	2.596	1379	2.727e−1
WATSON	31	1e−3	982	1.556e−1	1465	1.648e−1	4316	6.367e−3
WATSON	31	1e−5	4133	4.936e−3	2191	1.682e−2	3826	1.420e−3
WATSON	31	1e−7	13951	1.468e−4	4582	8.715e−4	8624	1.013e−4
WOODS	4	1e−1	86	8.655	378	5.200e−1	391	2.498e−1
WOODS	4	1e−3	87	7.873	300	1.818e−1	567	1.248e−5
WOODS	4	1e−5	432	1.680e−5	786	1.295e−3	513	6.023e−7
WOODS	4	1e−7	515	4.005e−7	278	3.224e−6	391	6.204e−10
WOODS	100	1e−1	8056	1.617e2	5272	4.980e1	5925	6.411
WOODS	100	1e−3	27605	4.700e−1	9371	4.582e−1	7344	1.666e−3
WOODS	100	1e−5	35366	1.415e−2	4207	4.455e−3	9097	9.510e−6
WOODS	100	1e−7	50000*	6.216e−2	4756	7.326e−5	7528	3.725e−7
ZANGWIL2	2	1e−1	7	1.455e−3	14	2.786e−2	31	1.009e−3
ZANGWIL2	2	1e−3	17	1.572e−5	13	1.557e−3	31	1.987e−6
ZANGWIL2	2	1e−5	21	6.543e−8	13	1.164e−6	516	3.655e−9
ZANGWIL2	2	1e−7	16	3.964e−10	13	1.167e−8	516	−9.150e−11

Table 21: Noisy Unconstrained `CUTEst` Problems Tested. n is the number of variables. σ_f is the standard deviation of the noise.

HS102, and HS103, KNITRO also hits the evaluation limit; however, it gets a better solution –a feasible one with a lower objective value– for all three problems.

For two problems, COBYLA terminated at a feasible point but KNITRO at an infeasible one. For problem POLAK6, KNITRO stalls at an infeasible point. For SPIRAL, it gets close to feasibility but runs out its evaluation budget before it can reduce the feasibility error further.

(iv) Both solvers terminated at infeasible points. For all three of those problems, KNITRO declares convergence to infeasible stationary points. For problems S365 and BURKEHAN, COBYLA terminates due to the condition on the final trust region radius whereas for POLAK2 it stops due to the function evaluation limit.

		LMDER		DFOLS	
Problem	(n, m)	#feval	$\phi(x) - \phi^*$	#feval	$\phi(x) - \phi^*$
LINEAR(FR)	$(9, 45)$	21	1.421e−14	45	−1.421e−14
LINEAR(FR)	$(9, 45)$	21	1.279e−13	49	0.000
LINEAR(R1)	$(7, 35)$	16	−3.099e−7	61	−3.099e−7
LINEAR(R1)	$(7, 35)$	16	−3.099e−7	56	−3.099e−7
LINEAR(R10RC)	$(7, 35)$	16	1.493e−8	56	1.493e−8
LINEAR(R10RC)	$(7, 35)$	16	1.493e−8	62	1.493e−8
ROSENBR	$(2, 2)$	39	0.000	50	3.788e−24
ROSENBR	$(2, 2)$	17	0.000	14	1.498e−20
HELIX	$(3, 3)$	49	3.941e−59	44	2.532e−28
HELIX	$(3, 3)$	57	5.921e−47	80	5.476e−20
POWELLSG	$(4, 4)$	331	6.610e−35	54	1.074e−18
POWELLSG	$(4, 4)$	346	7.106e−35	47	1.467e−14
FREUROTH	$(2, 2)$	60	3.688e−6	122	3.679e−6
FREUROTH	$(2, 2)$	87	3.793e−6	107	3.679e−6
BARD	$(3, 15)$	21	3.066e−10	32	3.066e−10
BARD	$(3, 15)$	169	1.742e1	107	1.066e−1
KOWOSB	$(4, 11)$	99	4.429e−12	77	3.849e−12
MEYER	$(3, 16)$	456	−4.829e−6	1500*	3.715e4
WATSON	$(6, 31)$	57	5.359e−11	81	5.355e−11
WATSON	$(6, 31)$	85	5.357e−11	105	5.355e−11
WATSON	$(9, 31)$	144	1.393e−13	409	1.468e−13
WATSON	$(9, 31)$	134	1.382e−13	499	1.454e−13
WATSON	$(12, 31)$	276	1.198e−15	340	2.214e−9
WATSON	$(12, 31)$	181	3.636e−16	1013	2.246e−9
BOX3D	$(3, 10)$	25	3.390e−32	52	1.921e−27
JENSMP	$(2, 10)$	46	−1.764e−5	70	−1.764e−5
BROWNDEN	$(4, 20)$	94	1.684e−3	167	1.626e−3
BROWNDEN	$(4, 20)$	96	1.733e−3	189	1.626e−3
CHEBYQUAD	$(6, 6)$	59	4.434e−32	54	1.239e−23
CHEBYQUAD	$(7, 7)$	58	6.915e−32	49	5.120e−29
CHEBYQUAD	$(8, 8)$	168	−2.725e−10	258	−2.743e−10
CHEBYQUAD	$(9, 9)$	94	4.143e−32	87	4.535e−28
CHEBYQUAD	$(10, 10)$	108	1.731e−3	191	−3.036e−10
CHEBYQUAD	$(11, 11)$	320	−4.428e−10	352	−4.481e−10
BROWNAL	$(10, 10)$	79	2.840e−29	52	7.676e−19
OSBORNE1	$(5, 33)$	157	−3.025e−12	783	3.737e−12
OSBORNE2	$(11, 65)$	101	−3.705e−9	150	−3.706e−9
OSBORNE2	$(11, 65)$	148	1.750	138	1.750
BDQRTIC	$(8, 8)$	508	3.759e−6	336	8.111e−6
BDQRTIC	$(10, 12)$	662	3.152e−6	457	3.874e−6
BDQRTIC	$(11, 14)$	1046	3.296e−6	554	2.046e−5
BDQRTIC	$(12, 16)$	1119	−1.268e−6	561	5.775e−5

Table 22: Benchmarking Problems Tested. n is the number of variables and m is the dimension of the residual vector.

		LMDER		DFOLS	
Problem	(n, m)	#feval	$\phi(x) - \phi^*$	#feval	$\phi(x) - \phi^*$
CUBE	$(5, 5)$	388	0.000	295	3.119e−18
CUBE	$(6, 6)$	1024	0.000	714	4.127e−24
CUBE	$(8, 8)$	4008^*	2.090e−8	4000^*	4.753e−11
MANCINO	$(5, 5)$	19	4.224e−22	25	6.176e−20
MANCINO	$(5, 5)$	25	4.686e−22	27	8.536e−18
MANCINO	$(8, 8)$	28	5.795e−22	23	3.152e−12
MANCINO	$(10, 10)$	34	8.218e−22	25	3.069e−11
MANCINO	$(12, 12)$	53	1.322e−22	32	2.454e−17
MANCINO	$(12, 12)$	53	5.201e−22	35	2.798e−9
HEART8LS	$(8, 8)$	37	3.402e−30	46	1.082e−23
HEART8LS	$(8, 8)$	109	3.402e−30	2293	1.001e−10

Table 23: Benchmarking Problems Tested. n is the number of variables and m is the dimension of the residual vector.

			LMDER		DFOLS	
Problem	(n, m)	σ_f	#feval	$\phi(x) - \phi^*$	#feval	$\phi(x) - \phi^*$
LINEAR(FR)	$(9, 45)$	1e−1	65	3.565e1	70	3.641e1
LINEAR(FR)	$(9, 45)$	1e−3	126	5.008e−2	97	6.606e−2
LINEAR(FR)	$(9, 45)$	1e−5	115	5.170e−4	81	2.889e−5
LINEAR(FR)	$(9, 45)$	1e−7	114	5.523e−6	76	2.296e−6
LINEAR(FR)	$(9, 45)$	1e−1	229	2.822e2	77	1.112e3
LINEAR(FR)	$(9, 45)$	1e−3	115	6.086e−2	89	2.179e−2
LINEAR(FR)	$(9, 45)$	1e−5	115	5.482e−4	78	2.305e−4
LINEAR(FR)	$(9, 45)$	1e−7	114	5.546e−6	76	2.240e−6
LINEAR(R1)	$(7, 35)$	1e−1	69	6.470e−2	74	3.081
LINEAR(R1)	$(7, 35)$	1e−3	65	3.221e−3	84	1.482e−4
LINEAR(R1)	$(7, 35)$	1e−5	70	−2.904e−7	74	−2.969e−7
LINEAR(R1)	$(7, 35)$	1e−7	108	−3.099e−7	73	−3.098e−7
LINEAR(R1)	$(7, 35)$	1e−1	68	6.471e−2	78	2.155
LINEAR(R1)	$(7, 35)$	1e−3	118	7.437e−5	86	1.577e−4
LINEAR(R1)	$(7, 35)$	1e−5	106	−3.049e−7	84	−2.926e−7
LINEAR(R1)	$(7, 35)$	1e−7	103	−3.099e−7	79	−3.099e−7
LINEAR(R10RC)	$(7, 35)$	1e−1	63	3.666e−1	71	8.280e−1
LINEAR(R10RC)	$(7, 35)$	1e−3	82	7.472e−6	82	2.771e−4
LINEAR(R10RC)	$(7, 35)$	1e−5	106	2.101e−8	75	2.978e−8
LINEAR(R10RC)	$(7, 35)$	1e−7	72	1.493e−8	73	1.493e−8
LINEAR(R10RC)	$(7, 35)$	1e−1	62	3.666e−1	79	9.209e−1
LINEAR(R10RC)	$(7, 35)$	1e−3	67	1.272e−3	89	2.258e−4
LINEAR(R10RC)	$(7, 35)$	1e−5	106	2.103e−8	82	1.660e−8
LINEAR(R10RC)	$(7, 35)$	1e−7	109	1.495e−8	75	1.985e−8
ROSENBR	$(2, 2)$	1e−1	44	7.600e−2	51	4.584e−1
ROSENBR	$(2, 2)$	1e−3	77	3.078e−7	50	6.040e−8
ROSENBR	$(2, 2)$	1e−5	67	4.501e−10	55	1.122e−10
ROSENBR	$(2, 2)$	1e−7	65	3.663e−14	49	2.148e−14
ROSENBR	$(2, 2)$	1e−1	105	4.118e−3	71	3.578e−2
ROSENBR	$(2, 2)$	1e−3	49	7.177e−7	35	6.097e−11
ROSENBR	$(2, 2)$	1e−5	44	4.539e−10	54	3.365e−10
ROSENBR	$(2, 2)$	1e−7	29	3.064e−14	28	2.191e−15
HELIX	$(3, 3)$	1e−1	61	4.158e−2	59	5.023e−2
HELIX	$(3, 3)$	1e−3	61	1.328e−6	62	1.265e−5
HELIX	$(3, 3)$	1e−5	71	7.012e−10	66	7.646e−10
HELIX	$(3, 3)$	1e−7	70	4.893e−14	77	2.031e−14
HELIX	$(3, 3)$	1e−1	61	1.767	66	8.701e−3
HELIX	$(3, 3)$	1e−3	75	5.707e−6	76	1.429e−6
HELIX	$(3, 3)$	1e−5	73	4.865e−10	74	8.032e−10
HELIX	$(3, 3)$	1e−7	70	5.016e−14	47	3.085e−16
POWELLSG	$(4, 4)$	1e−1	78	6.286e−2	62	9.436e−2
POWELLSG	$(4, 4)$	1e−3	82	3.495e−6	54	7.641e−5
POWELLSG	$(4, 4)$	1e−5	86	3.364e−9	54	7.255e−10
POWELLSG	$(4, 4)$	1e−7	131	2.803e−14	55	4.174e−15
POWELLSG	$(4, 4)$	1e−1	79	6.338e−2	59	2.036e−3
POWELLSG	$(4, 4)$	1e−3	86	3.404e−5	64	2.126e−7
POWELLSG	$(4, 4)$	1e−5	131	2.607e−10	61	1.502e−8
POWELLSG	$(4, 4)$	1e−7	121	1.694e−13	56	2.861e−15

Table 24: Benchmarking Problems Tested. n is the number of variables and m is the dimension of the residual vector.

			LMDER		DFOLS	
Problem	(n, m)	σ_f	#feval	$\phi(x) - \phi^*$	#feval	$\phi(x) - \phi^*$
FREUROTH	$(2, 2)$	1e−1	42	1.724	68	4.634e−2
FREUROTH	$(2, 2)$	1e−3	48	7.582e−3	75	4.555e−6
FREUROTH	$(2, 2)$	1e−5	59	7.704e−5	113	3.717e−6
FREUROTH	$(2, 2)$	1e−7	64	4.523e−6	91	3.679e−6
FREUROTH	$(2, 2)$	1e−1	55	2.546	83	1.310e−2
FREUROTH	$(2, 2)$	1e−3	66	1.265e−2	78	1.331e−5
FREUROTH	$(2, 2)$	1e−5	80	4.987e−5	102	3.681e−6
FREUROTH	$(2, 2)$	1e−7	91	7.027e−6	122	3.679e−6
BARD	$(3, 15)$	1e−1	45	2.370e−1	46	5.987e−2
BARD	$(3, 15)$	1e−3	41	1.168e−5	38	4.673e−5
BARD	$(3, 15)$	1e−5	53	4.713e−8	34	1.404e−9
BARD	$(3, 15)$	1e−7	44	8.436e−10	35	3.068e−10
BARD	$(3, 15)$	1e−1	41	1.355e1	56	9.256e−2
BARD	$(3, 15)$	1e−3	73	2.035e1	65	4.982e−5
BARD	$(3, 15)$	1e−5	98	2.282	54	1.114e−9
BARD	$(3, 15)$	1e−7	115	1.611e1	102	1.066e−1
KOWOSB	$(4, 11)$	1e−1	35	5.005e−3	42	3.088e−2
KOWOSB	$(4, 11)$	1e−3	57	1.418e−4	44	1.561e−4
KOWOSB	$(4, 11)$	1e−5	92	4.081e−8	60	4.916e−8
KOWOSB	$(4, 11)$	1e−7	84	1.302e−8	64	2.981e−10
MEYER	$(3, 16)$	1e−1	182	8.301e4	67	1.113e5
MEYER	$(3, 16)$	1e−3	666	4.770	252	7.380e4
MEYER	$(3, 16)$	1e−5	504	8.423e−3	1498	4.022e4
MEYER	$(3, 16)$	1e−7	497	−1.300e−6	1500*	4.031e4
WATSON	$(6, 31)$	1e−1	96	1.592e−1	71	1.232e−1
WATSON	$(6, 31)$	1e−3	83	1.818e−4	79	1.328e−4
WATSON	$(6, 31)$	1e−5	82	6.905e−5	82	7.375e−7
WATSON	$(6, 31)$	1e−7	103	5.333e−9	78	8.703e−10
WATSON	$(6, 31)$	1e−1	95	2.562e−1	80	1.150e−1
WATSON	$(6, 31)$	1e−3	236	4.267e−4	82	2.052e−4
WATSON	$(6, 31)$	1e−5	116	5.311e−5	87	1.676e−7
WATSON	$(6, 31)$	1e−7	116	2.197e−8	85	6.039e−10
WATSON	$(9, 31)$	1e−1	95	1.791e−1	95	6.326e−2
WATSON	$(9, 31)$	1e−3	183	2.761e−4	113	1.112e−4
WATSON	$(9, 31)$	1e−5	386	3.603e−5	114	5.717e−6
WATSON	$(9, 31)$	1e−7	531	1.175e−7	220	9.384e−10
WATSON	$(9, 31)$	1e−1	134	1.338	128	3.406e−2
WATSON	$(9, 31)$	1e−3	390	7.293e−3	115	2.543e−4
WATSON	$(9, 31)$	1e−5	580	1.436e−4	113	4.061e−6
WATSON	$(9, 31)$	1e−7	533	4.147e−7	337	5.464e−9
WATSON	$(12, 31)$	1e−1	147	1.829e−1	133	4.518e−1
WATSON	$(12, 31)$	1e−3	199	1.942e−4	129	4.431e−4
WATSON	$(12, 31)$	1e−5	736	1.863e−5	130	1.202e−7
WATSON	$(12, 31)$	1e−7	713	4.164e−8	234	1.337e−8
WATSON	$(12, 31)$	1e−1	188	7.230e−1	127	1.600e−1
WATSON	$(12, 31)$	1e−3	255	1.086e−2	139	2.802e−4
WATSON	$(12, 31)$	1e−5	530	3.237e−2	196	1.911e−8
WATSON	$(12, 31)$	1e−7	851	1.854e−4	183	1.560e−8

Table 25: Benchmarking Problems Tested. n is the number of variables and m is the dimension of the residual vector.

			LMDER		DFOLS	
Problem	(n, m)	σ_f	#feval	$\phi(x) - \phi^*$	#feval	$\phi(x) - \phi^*$
BOX3D	$(3, 10)$	1e−1	44	1.674e−1	53	1.181e−1
BOX3D	$(3, 10)$	1e−3	63	1.532e−3	43	5.394e−6
BOX3D	$(3, 10)$	1e−5	56	1.738e−9	46	5.818e−12
BOX3D	$(3, 10)$	1e−7	55	1.758e−13	57	2.248e−13
JENSMP	$(2, 10)$	1e−1	46	3.728e3	47	2.684e−1
JENSMP	$(2, 10)$	1e−3	44	4.102e−2	51	1.660e−2
JENSMP	$(2, 10)$	1e−5	53	1.445e−3	59	6.415e−5
JENSMP	$(2, 10)$	1e−7	51	−7.215e−6	47	−8.720e−6
BROWNDEN	$(4, 20)$	1e−1	75	1.521e3	84	3.922e2
BROWNDEN	$(4, 20)$	1e−3	86	3.277e1	97	2.897
BROWNDEN	$(4, 20)$	1e−5	98	7.498e−2	140	9.534e−3
BROWNDEN	$(4, 20)$	1e−7	105	2.529e−3	144	4.056e−3
BROWNDEN	$(4, 20)$	1e−1	90	2.384e3	102	8.007e2
BROWNDEN	$(4, 20)$	1e−3	102	4.493e1	97	3.732
BROWNDEN	$(4, 20)$	1e−5	117	4.912e−2	115	1.440e−1
BROWNDEN	$(4, 20)$	1e−7	114	2.743e−3	130	2.798e−3
CHEBYQUAD	$(6, 6)$	1e−1	54	4.243e−2	64	2.304e−2
CHEBYQUAD	$(6, 6)$	1e−3	57	3.020e−2	73	9.161e−7
CHEBYQUAD	$(6, 6)$	1e−5	93	4.322e−10	61	1.447e−11
CHEBYQUAD	$(6, 6)$	1e−7	91	4.785e−14	63	1.792e−13
CHEBYQUAD	$(7, 7)$	1e−1	46	3.377e−2	67	1.045e−2
CHEBYQUAD	$(7, 7)$	1e−3	118	6.182e−6	64	1.901e−5
CHEBYQUAD	$(7, 7)$	1e−5	97	9.566e−10	73	2.030e−10
CHEBYQUAD	$(7, 7)$	1e−7	95	1.043e−13	73	6.456e−14
CHEBYQUAD	$(8, 8)$	1e−1	69	2.683e−2	79	6.164e−2
CHEBYQUAD	$(8, 8)$	1e−3	93	1.261e−2	96	1.182e−3
CHEBYQUAD	$(8, 8)$	1e−5	128	1.039e−3	199	6.120e−7
CHEBYQUAD	$(8, 8)$	1e−7	240	6.147e−7	208	1.288e−9
CHEBYQUAD	$(9, 9)$	1e−1	54^{**}	2.888e−2	86	2.597e−2
CHEBYQUAD	$(9, 9)$	1e−3	100	2.013e−2	100	4.981e−5
CHEBYQUAD	$(9, 9)$	1e−5	134	2.326e−9	102	6.629e−10
CHEBYQUAD	$(9, 9)$	1e−7	161	3.856e−14	104	3.012e−14
CHEBYQUAD	$(10, 10)$	1e−1	94	4.026e−2	89	2.091e−2
CHEBYQUAD	$(10, 10)$	1e−3	153	9.237e−4	87	4.596e−3
CHEBYQUAD	$(10, 10)$	1e−5	163	5.114e−6	113	4.622e−5
CHEBYQUAD	$(10, 10)$	1e−7	207	1.732e−3	163	5.151e−8
CHEBYQUAD	$(11, 11)$	1e−1	78	2.983e−2	87	1.920e−2
CHEBYQUAD	$(11, 11)$	1e−3	104	9.225e−3	93	6.273e−3
CHEBYQUAD	$(11, 11)$	1e−5	213	4.205e−4	235	8.670e−7
CHEBYQUAD	$(11, 11)$	1e−7	449	1.141e−6	246	1.473e−8

Table 26: Benchmarking Problems Tested. n is the number of variables and m is the dimension of the residual vector.

			LMDER		DFOLS	
Problem	(n, m)	σ_f	#feval	$\phi(x) - \phi^*$	#feval	$\phi(x) - \phi^*$
BROWNAL	$(10, 10)$	1e−1	127	1.419	89	1.655e2
BROWNAL	$(10, 10)$	1e−3	147	5.220e−7	99	3.284e−5
BROWNAL	$(10, 10)$	1e−5	155	4.102e−10	89	9.707e−10
BROWNAL	$(10, 10)$	1e−7	131	1.084e−13	75	1.308e−14
OSBORNE1	$(5, 33)$	1e−1	53	1.279	53	5.737
OSBORNE1	$(5, 33)$	1e−3	75	3.424e−3	86	4.512e−1
OSBORNE1	$(5, 33)$	1e−5	86	1.010e−2	225	2.446e−2
OSBORNE1	$(5, 33)$	1e−7	376	4.951e−8	185	2.446e−2
OSBORNE2	$(11, 65)$	1e−1	113	1.297	104	2.226
OSBORNE2	$(11, 65)$	1e−3	426	6.798e−3	153	2.216e−4
OSBORNE2	$(11, 65)$	1e−5	308	9.726e−5	174	6.715e−7
OSBORNE2	$(11, 65)$	1e−7	235	1.654e−7	148	8.159e−10
OSBORNE2	$(11, 65)$	1e−1	124	2.799e1	111	2.293e1
OSBORNE2	$(11, 65)$	1e−3	204	1.754	132	1.750
OSBORNE2	$(11, 65)$	1e−5	277	1.750	134	1.750
OSBORNE2	$(11, 65)$	1e−7	223	1.750	139	1.750
BDQRTIC	$(8, 8)$	1e−1	124	5.023	89	4.566
BDQRTIC	$(8, 8)$	1e−3	128	2.317	103	2.289e−1
BDQRTIC	$(8, 8)$	1e−5	265	2.905e−3	122	1.864e−2
BDQRTIC	$(8, 8)$	1e−7	437	2.902e−5	173	8.502e−5
BDQRTIC	$(10, 12)$	1e−1	128	1.221e1	97	1.447e2
BDQRTIC	$(10, 12)$	1e−3	197	1.021e−1	124	1.141
BDQRTIC	$(10, 12)$	1e−5	340	5.836e−3	153	3.739e−2
BDQRTIC	$(10, 12)$	1e−7	574	6.811e−5	262	7.248e−4
BDQRTIC	$(11, 14)$	1e−1	139	7.006	106	9.772e1
BDQRTIC	$(11, 14)$	1e−3	188	1.299e−1	127	3.141
BDQRTIC	$(11, 14)$	1e−5	396	6.226e−3	190	7.543e−2
BDQRTIC	$(11, 14)$	1e−7	725	4.880e−5	333	2.513e−3
BDQRTIC	$(12, 16)$	1e−1	137	5.551	125	1.355e2
BDQRTIC	$(12, 16)$	1e−3	204	2.591e−1	135	2.972
BDQRTIC	$(12, 16)$	1e−5	283	1.565e−2	191	5.335e−2
BDQRTIC	$(12, 16)$	1e−7	626	3.785e−4	323	3.455e−3
CUBE	$(5, 5)$	1e−1	96	2.386e−2	54	6.129e−2
CUBE	$(5, 5)$	1e−3	112	3.481e−3	90	1.519e−4
CUBE	$(5, 5)$	1e−5	250	1.971e−10	367	4.920e−12
CUBE	$(5, 5)$	1e−7	415	9.699e−14	272	1.508e−13
CUBE	$(6, 6)$	1e−1	127	1.312e−2	56	5.761e−2
CUBE	$(6, 6)$	1e−3	162	1.853e−4	82	1.329e−4
CUBE	$(6, 6)$	1e−5	832	8.835e−7	460	2.319e−6
CUBE	$(6, 6)$	1e−7	958	9.762e−14	758	4.059e−14
CUBE	$(8, 8)$	1e−1	168	2.402e−1	63	9.869e1
CUBE	$(8, 8)$	1e−3	185	8.640e−6	87	7.661e−4
CUBE	$(8, 8)$	1e−5	621	1.251e−5	263	1.030e−4
CUBE	$(8, 8)$	1e−7	3431	6.217e−8	1763	2.522e−7

Table 27: Benchmarking Problems Tested. n is the number of variables and m is the dimension of the residual vector.

			LMDER		DFOLS	
Problem	(n, m)	σ_f	#feval	$\phi(x) - \phi^*$	#feval	$\phi(x) - \phi^*$
MANCINO	$(5, 5)$	1e−1	47	7.329e−2	48	1.334e−2
MANCINO	$(5, 5)$	1e−3	45	7.341e−6	44	5.405e−8
MANCINO	$(5, 5)$	1e−5	39	7.341e−10	33	1.657e−12
MANCINO	$(5, 5)$	1e−7	39	7.342e−14	25	3.807e−15
MANCINO	$(5, 5)$	1e−1	47	7.435e−2	58	2.132e−2
MANCINO	$(5, 5)$	1e−3	45	7.511e−6	44	1.457e−5
MANCINO	$(5, 5)$	1e−5	45	1.247e−9	27	4.630e−10
MANCINO	$(5, 5)$	1e−7	45	8.751e−14	27	4.120e−14
MANCINO	$(8, 8)$	1e−1	72	1.550e−1	82	1.228e−1
MANCINO	$(8, 8)$	1e−3	78	1.130e−7	57	8.045e−6
MANCINO	$(8, 8)$	1e−5	60	1.549e−9	46	2.898e−10
MANCINO	$(8, 8)$	1e−7	60	1.549e−13	23	7.222e−13
MANCINO	$(10, 10)$	1e−1	109	8.335e−5	82	7.748e−3
MANCINO	$(10, 10)$	1e−3	107	8.191e−9	59	8.847e−6
MANCINO	$(10, 10)$	1e−5	74	6.522e−10	45	1.015e−10
MANCINO	$(10, 10)$	1e−7	74	6.526e−14	36	5.372e−15
MANCINO	$(12, 12)$	1e−1	114	3.374e−4	107	3.447e−2
MANCINO	$(12, 12)$	1e−3	101	3.354e−8	77	8.807e−6
MANCINO	$(12, 12)$	1e−5	101	3.353e−12	60	9.663e−10
MANCINO	$(12, 12)$	1e−7	101	3.357e−16	32	9.721e−14
MANCINO	$(12, 12)$	1e−1	114	3.374e−4	126	9.117e−2
MANCINO	$(12, 12)$	1e−3	114	3.348e−8	72	9.369e−6
MANCINO	$(12, 12)$	1e−5	101	3.357e−12	35	4.632e−9
MANCINO	$(12, 12)$	1e−7	101	3.452e−16	35	2.607e−9
HEART8LS	$(8, 8)$	1e−1	113	6.289e−2	71	1.356e−1
HEART8LS	$(8, 8)$	1e−3	83	1.244e−6	66	3.717e−6
HEART8LS	$(8, 8)$	1e−5	82	1.508e−11	66	3.928e−12
HEART8LS	$(8, 8)$	1e−7	79	1.167e−15	67	1.275e−14
HEART8LS	$(8, 8)$	1e−1	156	5.632	86	5.270
HEART8LS	$(8, 8)$	1e−3	566	7.222e−6	77	4.911
HEART8LS	$(8, 8)$	1e−5	164	1.102e−9	172	4.794
HEART8LS	$(8, 8)$	1e−7	161	5.788e−14	376	4.428

Table 28: Benchmarking Problems Tested. n is the number of variables and m is the dimension of the residual vector.

Problem	Objective Type	Number of Variables (n)	One-sided Bounds	Two-sided Bounds	Number of Constraints (m)	Linear Ineq.	Nonlinear Ineq.
BURKEHAN	quadratic	1	1	0	1	0	1
CANTILVR	linear	5	5	0	1	0	1
CB2	linear	3	0	0	3	0	3
CB3	linear	3	0	0	3	0	3
CHACONN1	linear	3	0	0	3	0	3
CHACONN2	linear	3	0	0	3	0	3
CRESC4	general	6	3	1	8	0	8
CRESC50	general	6	3	1	100	0	100
DEMBO7	quadratic	16	0	16	20	1	19
DIPIGRI	general	7	0	0	4	0	4
GIGOMEZ2	linear	3	0	0	3	0	3
GIGOMEZ3	linear	3	0	0	3	0	3
HALDMADS	linear	6	0	0	42	0	42
HS100	general	7	0	0	4	0	4
HS100MOD	general	7	0	0	4	0	4
HS101	general	7	0	7	5	0	5
HS102	general	7	0	7	5	0	5
HS103	general	7	0	7	5	0	5
HS104	general	8	0	8	5	0	5
HS13	quadratic	2	2	0	1	0	1
HS34	linear	3	0	3	2	0	2
HS64	general	3	3	0	1	0	1
HS66	linear	3	0	3	2	0	2
HS67	general	3	0	3	14	0	14
HS72	linear	4	0	4	2	0	2
HS85	general	5	0	5	21	1	20
HS88	quadratic	2	0	0	1	0	1
HS89	quadratic	3	0	0	1	0	1
HS90	quadratic	4	0	0	1	0	1
HS91	quadratic	5	0	0	1	0	1
HS92	quadratic	6	0	0	1	0	1
HS93	general	6	6	0	2	0	2
MADSEN	linear	3	0	0	6	0	6
MATRIX2	quadratic	6	4	0	2	0	2
MINMAXBD	linear	5	0	0	20	0	20
POLAK1	linear	3	0	0	2	0	2
POLAK2	linear	11	0	0	2	0	2
POLAK3	linear	12	0	0	10	0	10
POLAK5	linear	3	0	0	2	0	2
POLAK6	linear	5	0	0	4	0	4
S365	quadratic	7	4	0	5	0	5
S365MOD	quadratic	7	4	0	5	0	5
SNAKE	linear	2	0	0	2	0	2
SPIRAL	linear	3	0	0	2	0	2
SYNTHES1	general	6	0	6	6	4	2
TWOBARS	general	2	0	2	2	0	2
WOMFLET	linear	3	0	0	3	0	3

Table 29: Properties of small-scale `CUTEst` problems.

Problem	SIF Parameter	Testing Name	d	m
SVANBERG	$N = 10$	SVANBERGN10	10	10
SVANBERG	$N = 50$	SVANBERGN50	50	50
SVANBERG	$N = 100$	SVANBERGN100	100	100
SVANBERG	$N = 500$	SVANBERGN500	500	500
READING4	$N = 2$	READING4N2	2	2
READING4	$N = 50$	READING4N50	50	50
READING4	$N = 100$	READING4N100	100	100
READING4	$N = 500$	READING4N500	500	500
COSHFUN	$M = 3$	COSHFUNM3	10	3
COSHFUN	$M = 8$	COSHFUNM8	25	8
COSHFUN	$M = 14$	COSHFUNM14	43	14
COSHFUN	$M = 20$	COSHFUNM20	61	20

Table 30: Instances of variable-size `CUTEst` problems.

	KNITRO				COBYLA			
Problem	$\phi(x)$	#feval	CPU	feaserr	$\phi(x)$	#fevals	CPU	feaserr
CB3	2.0000	40	0.023	2.22e-16	2.0000	44	0.157	0.00e+00
CHACONN2	2.0000	40	0.021	0.00e+00	2.0000	48	0.156	0.00e+00
GIGOMEZ3	2.0000	40	0.022	0.00e+00	2.0000	45	0.160	0.00e+00
HS100	680.6301	360	0.160	0.00e+00	680.6300	454	0.272	0.00e+00
DIPIGRI	680.6301	389	0.152	2.84e-14	680.6300	458	0.288	0.00e+00
HS93	135.0760	2247	1.040	4.44e-16	135.0760	3000*	0.873	0.00e+00
HS64	6299.8424	144	0.093	0.00e+00	6299.8400	409	0.244	0.00e+00
POLAK3	5.9330	902	0.414	3.89e-16	5.9330	3139	0.071	0.00e+00
POLAK1	2.7183	60	0.034	0.00e+00	2.7183	326	0.232	0.00e+00
HS104	3.9512	718	0.282	0.00e+00	3.9512	3977	0.131	0.00e+00
HS100MOD	678.6796	477	0.207	0.00e+00	678.6790	385	0.240	0.00e+00
TWOBARS	1.5087	63	0.037	0.00e+00	1.5087	83	0.172	0.00e+00
HS85	-2.2156	117	0.078	7.11e-15	-2.2156	299	0.242	0.00e+00
CB2	1.9522	72	0.041	1.11e-16	1.9522	111	0.173	0.00e+00
CHACONN1	1.9522	55	0.030	0.00e+00	1.9522	120	0.175	0.00e+00
MADSEN	0.6164	73	0.042	0.00e+00	0.6164	112	0.175	0.00e+00
HS66	0.5182	43	0.026	4.44e-16	0.5182	101	0.170	0.00e+00
MINMAXBD	115.7064	420	0.335	6.17e-14	115.7060	924	0.395	0.00e+00
CANTILVR	1.3400	170	0.078	3.61e-16	1.3400	281	0.227	0.00e+00
HS92	1.3627	976	0.988	4.60e-17	1.3627	3000*	0.993	0.00e+00
HALDMADS	0.0001	113	0.073	4.44e-16	0.0001	669	0.341	0.00e+00
HS34	-0.8340	59	0.033	4.44e-16	-0.8340	47	0.149	0.00e+00
HS90	1.3627	570	0.475	0.00e+00	1.3629	2000*	0.066	0.00e+00
SNAKE	0.0000	25	0.029	1.02e-16	0.0000	72	0.165	0.00e+00
HS72	727.6794	108	0.103	1.73e-18	727.6794	1002	0.424	4.77e-18
GIGOMEZ2	1.9522	79	0.048	2.22e-16	1.9522	111	0.169	1.11e-16
SYNTHES1	0.7593	96	0.041	0.00e+00	0.7593	174	0.186	1.54e-23
HS88	1.3627	209	0.156	0.00e+00	1.3627	167	0.203	2.09e-17
HS13	0.9999	140	0.155	2.61e-13	1.0000	86	0.168	1.07e-27
MATRIX2	0.0000	275	0.131	3.60e-17	0.0000	216	0.199	5.20e-18
WOMFLET	0.0000	103	0.070	5.89e-10	0.0000	117	0.181	8.99e-18
S365	0.0000	63	0.032	1.72e-07	0.0000	210	0.215	4.35e-17
HS67	-1162.1187	285	0.198	0.00e+00	-980.1600	7000*	0.885	0.00e+00
CRESC50	0.5952	16704	17.912	0.00e+00	7.9917	50000*	4.863	0.00e+00
CRESC4	0.8719	688	0.363	0.00e+00	44.5901	4000*	0.318	0.00e+00
HS91	1.3627	1229	1.057	9.96e-18	1.1199	2500*	0.583	6.50e-04
DEMBO7	174.7870	3225	1.594	3.61e-16	250.0720	10000*	0.238	7.29e-02
POLAK5	50.0000	65	0.043	0.00e+00	34.4929	1500*	0.571	1.55e+01
HS101	1809.7691	3500*	1.908	1.38e-13	3000.1900	3500*	0.077	1.99e-01
HS89	1.3627	315	0.257	2.17e-17	0.6520	1500*	0.742	5.12e-02
HS102	911.8816	3500*	1.965	4.67e-12	3000.3500	3500*	0.096	3.60e-01
HS103	543.6680	3500*	1.858	2.57e-16	3000.1700	3500*	0.091	1.76e-01
SPIRAL	0.0195	1500*	0.796	1.34e-05	0.0958	1500*	0.508	0.00e+00
POLAK6	-78.6745	474	1.311	8.55e+01	0.0000	2500*	0.765	0.00e+00
S365MOD	0.2500	162	0.204	1.25e+00	0.0301	247	0.211	3.26e-01
BURKEHAN	10.0000	13	0.007	1.01e+02	0.0000	54	0.175	1.00e+00
POLAK2	29.9570	2641	5.119	5.39e+01	-259.2500	5500*	0.794	3.14e+02

Table 31: Summary of the results for small-scale noiseless constrained `CUTEst` problems. The horizontal bars divide cases (i), (ii), (iii), and (iv).

		$TOL = 10^{-6}$					$TOL = 10^{-3}$					$TOL = 10^{-1}$				
			KNITRO		COBYLA			KNITRO		COBYLA			KNITRO		COBYLA	
problem	ϕ^*	target ϕ	$\phi(x)$	#fevals	$\phi(x)$	#fevals	target ϕ	$\phi(x)$	#fevals	$\phi(x)$	#fevals	target ϕ	$\phi(x)$	#fevals	$\phi(x)$	#fevals
CB3	2.000000	2.00000	2.00000	30	2.00000	25	2.00200	2.00000	30	2.00000	25	2.20000	2.00000	30	2.00351	18
CHACONN2	2.000000	2.00000	2.00000	30	2.00000	32	2.00200	2.00000	30	2.00021	24	2.20000	2.00000	30	2.06187	19
GIGOMEZ3	2.000000	2.00000	2.00000	30	2.00000	39	2.00200	2.00000	30	2.00000	39	2.20000	2.00000	30	2.00000	39
HS100	680.630057	680.63074	680.63013	124	680.63000	247	681.31069	680.75511	75	680.63000	247	748.69306	714.00000	1	714.00000	1
DIPIGRI	680.630057	680.63074	680.63013	124	680.63000	410	681.31069	680.75511	75	680.63100	141	748.69306	714.00000	1	714.00000	1
HS93	135.075964	135.07610	135.07610	871	135.07600	2416	135.21104	135.14058	229	135.21000	270	148.58356	137.06640	1	137.06600	1
HS64	6299.842428	6299.84873	6299.73981	64	6299.84000	335	6306.14227	6302.48265	54	6305.80000	316	6929.82667	6687.88225	30	6684.54000	239
POLAK3	5.933003	5.93301	5.93301	469	5.93300	2637	5.93894	5.93353	349	5.93644	2350	6.52630	5.93353	349	5.94016	2328
POLAK1	2.718282	2.71828	2.71828	50	2.71828	265	2.72100	2.71828	50	2.71834	253	2.99011	2.71828	50	2.83811	250
HS104	3.951163	3.95117	3.95117	290	3.95116	1764	3.95511	3.95148	231	3.95381	730	4.34628	3.95946	193	4.15766	173
HS100MOD	678.679638	678.68032	678.67969	173	678.68000	169	679.35832	679.08174	63	679.18500	50	746.54760	714.00000	1	714.00000	1
TWOBARS	1.508652	1.50865	1.50865	50	1.50865	33	1.51016	1.50865	50	1.50892	21	1.65952	1.50865	50	1.65447	10
HS85	-2.215605	-2.21560	-2.21560	110	-2.21560	299	-2.21339	-2.21560	110	-2.21500	247	-1.99404	-2.21560	110	-2.02180	158
CB2	1.952224	1.95223	1.95222	52	1.95222	57	1.95418	1.95222	52	1.95230	41	2.14745	1.95222	52	1.95230	41
CHACONN1	1.952224	1.95223	1.95222	40	1.95222	55	1.95418	1.95222	40	1.95222	55	2.14745	1.95222	40	2.00282	13
MADSEN	0.616432	0.61643	0.61643	58	0.61643	65	0.61743	0.61643	58	0.61643	65	0.71643	0.61643	58	0.65433	22
HS66	0.518163	0.51816	0.51816	33	0.51816	37	0.51916	0.51866	21	0.51817	28	0.61816	0.58000	1	0.58000	1
MINMAXBD	115.706440	115.70656	115.70644	343	115.70600	800	115.82215	115.70644	343	115.72100	766	127.27708	118.74188	246	115.72100	766
CANTILVR	1.339956	1.33996	1.33994	107	1.33995	189	1.34130	1.33994	107	1.33997	130	1.47395	1.33994	107	1.34758	107
HS92	1.362657	1.36266	1.36266	844	1.36265	2121	1.36402	1.36327	405	1.36310	203	1.49892	1.36327	405	1.36310	203
HALDMADS	0.000122	0.00012	0.00012	97	0.00013	669	0.00112	0.00012	97	0.00057	51	0.10012	0.00012	97	0.00057	51
HS34	-0.834032	-0.83403	-0.83403	54	-0.83400	40	-0.83303	-0.83403	54	-0.83400	25	-0.73403	-0.83186	44	-0.83400	25
HS90	1.362657	1.36266	1.36185	259	1.36287	2000	1.36402	1.36185	259	1.36311	159	1.49892	1.36185	259	1.36311	159
SNAKE	0.000000	0.00000	0.00000	17	0.00000	67	0.00100	0.00000	17	0.00009	61	0.10000	0.00000	17	0.00009	61
HS72	727.679358	727.68009	727.66249	72	727.67973	877	728.40704	727.66249	72	728.36304	856	800.44729	727.66249	72	738.77714	832
GIGOMEZ2	1.952224	1.95223	1.95222	64	1.95223	47	1.95418	1.95222	64	1.95266	32	2.14745	1.95222	64	2.08642	13
SYNTHES1	0.759284	0.75929	0.75928	72	0.75928	79	0.76028	0.75928	72	0.75928	79	0.85928	0.75928	72	0.79837	17
HS88	1.362657	1.36266	1.36249	189	1.36266	138	1.36402	1.36249	189	1.36387	123	1.49892	1.36249	189	1.36387	123
HS13	0.999872	0.99987	0.99987	116	1.00000	86	1.00087	1.00060	88	1.00085	41	1.09987	1.07956	40	1.07745	22
MATRIX2	0.000000	0.00000	0.00000	131	0.00000	187	0.00100	0.00001	107	0.00000	187	0.10000	0.00001	107	0.09622	46
WOMFLET	0.000000	0.00000	0.00000	78	0.00000	104	0.00100	0.00000	78	0.00000	104	0.10000	0.00000	78	0.00000	104
S365	0.000000	0.00000	0.00000	63	0.00000	179	0.00100	0.00000	63	0.00000	179	0.10000	0.00000	63	0.00000	179

Table 32: Function evaluations to obtain $\phi_k \leq \phi^* + TOL \cdot \max\{1.0, |\phi^*|\}$ for small scale noiseless constrained problems.

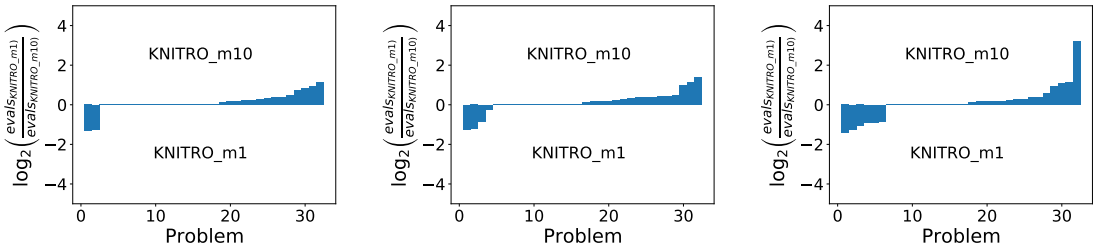

Figure 29: *Efficiency, Noiseless Case.* Log-ratio profiles comparing KNITRO with an L-BFGS Hessian approximation of memory one and memory 10 when $\epsilon(x) = 0$. The figures measure number of function evaluations to satisfy (4.7) for $\tau = 10^{-1}$ (left), 10^{-3} (middle), 10^{-6} (right).

C.3.3 Noisy Functions

In this section, we list the final objective value($\phi(x)$), feasibility error(feaserr) and the number of function evaluations (#feval) needed to achieve this for each problem instance, varying the noise level $\sigma_f \in \{10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}\}$, in Tables 33-38. Function evaluations marked with a * denote cases where the algorithm reached the maximum number of function evaluations. An 'f' marks the cases where the feasibility error is large compared to the noise level, i.e., $\|\max\{\psi(x), 0\}\|_\infty \geq \sqrt{3\sigma_f}$.

	KNITRO					COBYLA				
σ_f	0.0	1e-07	1e-05	1e-03	1e-01	0.0	1.00e-07	1.00e-05	1.00e-03	1.00e-01
CB3	4.92e-12	1.00e-08	2.00e-06	1.35e-04	1.51e-01	8.36e-12	1.00e-08	6.00e-06	1.10e-03	2.97e-01
CHACONN2	1.72e-12	1.00e-08	1.60e-05	3.80e-04	8.27e-02	1.72e-12	1.00e-08	3.00e-06	1.01e-03	f
GIGOMEZ3	9.80e-13	1.00e-08	2.00e-06	1.35e-04	1.62e-01	9.80e-13	1.00e-08	1.00e-05	9.30e-05	f
HS100	2.05e-12	2.37e-06	2.14e-04	1.34e-02	2.16e+00	5.87e-07	1.63e-06	9.62e-04	3.13e-02	3.65e-02
DIPIGRI	4.09e-12	3.24e-05	1.87e-04	1.25e-02	1.98e+00	2.05e-12	1.63e-06	4.23e-04	3.21e-02	9.14e-01
HS93	3.90e-09	4.34e-03	6.21e-02	2.49e-01	2.69e+00	3.96e-05	1.52e-01	4.88e-01	1.79e+00	1.99e+00
HS64	2.28e-09	4.29e-05	3.10e-03	4.90e+00	8.78e+02	2.28e-09	2.62e-04	7.11e-02	f	f
POLAK3	1.00e-12	6.51e-07	2.13e-04	4.39e-03	f	1.01e-12	5.65e-06	3.58e-04	4.13e-04	f
POLAK1	1.00e-12	1.72e-07	6.17e-06	1.53e-03	2.62e+00	9.90e-13	1.72e-07	1.17e-06	3.40e-02	5.15e+00
HS104	1.05e-11	5.36e-05	1.30e-02	2.99e-02	6.63e-01	1.05e-11	1.55e-03	1.38e-02	1.37e-01	2.74e+00
HS100MOD	1.02e-12	2.01e-05	2.48e-04	2.06e-02	1.34e+00	1.02e-12	1.11e-07	2.13e-04	8.63e-03	1.20e+00
TWOBARS	1.51e-12	4.18e-07	1.14e-05	1.39e-04	9.27e-02	1.51e-12	4.18e-07	1.54e-04	1.56e-03	5.86e-02
HS85	6.00e-14	3.12e-07	3.12e-07	5.72e-03	9.27e-01	4.51e-06	3.12e-07	6.88e-07	4.82e-01	9.59e-01
CB2	3.70e-13	5.06e-07	4.49e-06	5.47e-04	2.05e-01	3.70e-13	4.94e-07	3.49e-06	8.49e-06	f
CHACONN1	8.80e-13	4.94e-07	6.49e-06	3.96e-03	1.34e-01	1.07e-12	4.94e-07	1.51e-06	6.81e-04	2.44e-02
MADSEN	1.00e-12	5.64e-07	7.56e-06	1.11e-04	2.58e-01	1.01e-12	5.64e-07	6.44e-06	2.90e-04	2.66e-01
HS66	8.64e-13	2.74e-07	1.37e-05	6.56e-04	2.53e-02	8.64e-13	2.74e-07	4.27e-06	5.23e-04	9.11e-03
MINMAXBD	1.01e-12	3.21e-07	5.63e-05	3.75e-03	f	2.52e-10	3.21e-07	2.03e-05	1.65e-04	1.31e+01
CANTILVR	4.40e-13	3.61e-07	1.64e-06	1.72e-03	1.91e-01	4.40e-13	6.39e-07	7.16e-05	8.57e-03	f
HS92	1.06e-09	1.16e-02	1.18e-01	f	9.97e-01	1.40e-07	5.02e-03	2.51e-01	2.14e+00	1.07e+00
HALDMADS	9.94e-13	3.71e-07	1.06e-05	7.51e-03	2.53e-01	9.52e-06	3.66e-05	5.66e-05	4.94e-04	f
HS34	2.89e-13	4.45e-07	2.55e-06	4.24e-04	6.93e-01	3.92e-09	4.45e-07	1.24e-05	1.82e-03	8.34e-01
HS90	1.06e-09	1.09e-04	6.99e-01	f	5.73e-01	2.18e-04	7.24e-03	3.13e-01	1.30e+00	1.03e+00
SNAKE	7.96e-13	f	2.28e+00	4.51e+01	8.35e+00	6.74e-07	2.12e-02	3.24e-02	3.56e-02	1.28e+00
HS72	0.00e+00	6.83e-02	2.53e+01	3.78e+02	f	0.00e+00	1.63e-02	f	f	f
GIGOMEZ2	0.00e+00	4.94e-07	1.15e-05	3.21e-04	7.78e-02	0.00e+00	4.94e-07	2.95e-05	9.14e-04	1.04e-01
SYNTHES1	0.00e+00	2.39e-06	7.39e-06	6.80e-03	1.22e+00	4.00e-15	2.61e-06	7.98e-02	8.49e-02	1.10e+00
HS88	1.10e-13	1.29e-04	3.85e-02	4.04e-01	1.32e+00	4.08e-12	1.76e-04	1.82e-01	7.87e-01	1.13e+00
HS13	2.97e-08	1.08e-02	4.54e-02	1.43e-01	6.42e-01	1.28e-04	6.70e-03	2.95e-02	1.15e-01	1.38e+00
MATRIX2	7.93e-16	1.00e-08	1.10e-05	3.63e-03	7.53e-02	3.82e-17	1.00e-08	3.80e-05	2.52e-03	1.19e+00
WOMFLET	7.89e-13	2.90e-05	1.39e-03	8.39e-03	6.25e+00	7.89e-13	1.00e-08	2.00e-06	3.12e-03	3.34e+00
S365	0.00e+00	1.00e-08	1.00e-08	1.00e-08	1.00e-08	8.89e-35	1.00e-08	1.00e-08	1.00e-08	1.00e-08

Table 33: Optimality gap $|\phi(x_k) - \phi^*|$ for small scale noisy constrained `CUTEst` problems.

	KNITRO					COBYLA				
σ_f	0.0	1e-07	1e-05	1e-03	1e-01	0.0	1.00e-07	1.00e-05	1.00e-03	1.00e-01
CB3	2.22e-16	0.00e+00	0.00e+00	7.50e-05	0.00e+00	0.00e+00	0.00e+00	1.20e-05	0.00e+00	3.71e-02
CHACONN2	0.00e+00	0.00e+00	0.00e+00	1.22e-04	2.54e-01	0.00e+00	0.00e+00	5.00e-06	0.00e+00	1.53e+00
GIGOMEZ3	0.00e+00	0.00e+00	0.00e+00	7.50e-05	0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.32e-04	5.89e-01
HS100	0.00e+00	5.00e-06	2.10e-05	6.18e-04	1.01e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.67e-01
DIPIGRI	2.84e-14	3.10e-05	3.00e-06	5.53e-04	1.38e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
HS93	4.44e-16	0.00e+00	1.00e-05	1.03e-03	1.13e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
HS64	0.00e+00	0.00e+00	1.00e-05	2.18e-03	4.96e-01	0.00e+00	0.00e+00	0.00e+00	1.82e-01	4.97e+00
POLAK3	3.89e-16	1.00e-06	8.00e-06	1.34e-03	5.56e-01	0.00e+00	0.00e+00	0.00e+00	8.60e-03	8.72e+00
POLAK1	0.00e+00	0.00e+00	0.00e+00	2.54e-04	6.74e-02	0.00e+00	0.00e+00	1.10e-05	9.15e-04	9.40e-02
HS104	0.00e+00	0.00e+00	0.00e+00	1.04e-03	2.81e-01	0.00e+00	0.00e+00	3.00e-06	2.80e-02	3.70e-01
HS100MOD	0.00e+00	0.00e+00	2.80e-05	3.91e-04	1.77e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	7.18e-02
TWOBARS	0.00e+00	0.00e+00	9.00e-06	1.28e-03	1.86e-02	0.00e+00	0.00e+00	0.00e+00	1.04e-03	1.40e-01
HS85	7.11e-15	1.00e-06	7.00e-06	1.52e-04	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.41e-03	0.00e+00
CB2	1.11e-16	0.00e+00	1.40e-05	0.00e+00	1.15e-02	0.00e+00	0.00e+00	6.00e-06	1.02e-03	1.10e+00
CHACONN1	0.00e+00	1.00e-06	1.40e-05	0.00e+00	2.43e-03	0.00e+00	0.00e+00	2.00e-06	7.34e-04	2.38e-02
MADSEN	0.00e+00	0.00e+00	0.00e+00	4.89e-04	0.00e+00	0.00e+00	0.00e+00	1.00e-05	3.03e-04	0.00e+00
HS66	4.44e-16	0.00e+00	3.00e-06	0.00e+00	0.00e+00	0.00e+00	0.00e+00	5.00e-06	1.21e-03	4.57e-02
MINMAXBD	6.17e-14	2.20e-05	1.90e-05	2.96e-04	4.65e+02	0.00e+00	0.00e+00	0.00e+00	1.53e-03	0.00e+00
CANTILVR	3.61e-16	0.00e+00	8.00e-06	0.00e+00	5.74e-02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	2.47e+00
HS92	4.60e-17	0.00e+00	2.58e-04	1.33e-01	1.35e-01	0.00e+00	0.00e+00	6.73e-04	1.04e-03	7.11e-02
HALDMADS	4.44e-16	2.00e-06	1.40e-05	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.50e-05	9.94e-04	1.82e+01
HS34	4.44e-16	0.00e+00	6.00e-06	0.00e+00	1.15e-01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
HS90	0.00e+00	0.00e+00	1.20e-05	1.33e-01	1.40e-01	0.00e+00	0.00e+00	1.07e-03	6.30e-05	1.20e-01
SNAKE	1.02e-16	1.70e-03	2.12e-04	2.90e-03	1.87e-01	0.00e+00	1.00e-06	3.00e-06	3.27e-04	3.65e-01
HS72	1.73e-18	0.00e+00	7.31e-04	4.62e-02	5.76e-01	4.77e-18	0.00e+00	7.94e-03	3.47e-01	3.26e+00
GIGOMEZ2	2.22e-16	2.00e-06	0.00e+00	6.68e-04	1.78e-01	1.11e-16	0.00e+00	0.00e+00	1.29e-03	0.00e+00
SYNTHES1	0.00e+00	1.00e-06	5.00e-06	0.00e+00	2.09e-01	1.54e-23	0.00e+00	0.00e+00	0.00e+00	0.00e+00
HS88	0.00e+00	0.00e+00	4.40e-05	2.05e-03	1.26e-01	2.09e-17	0.00e+00	3.57e-04	1.82e-02	1.47e-01
HS13	2.61e-13	0.00e+00	1.20e-05	4.07e-04	6.52e-02	1.07e-27	0.00e+00	3.00e-06	0.00e+00	0.00e+00
MATRIX2	3.60e-17	0.00e+00	9.00e-06	7.30e-04	2.95e-02	5.20e-18	0.00e+00	6.00e-06	3.17e-04	0.00e+00
WOMFLET	5.89e-10	1.50e-05	1.00e-06	4.53e-04	5.05e-02	8.99e-18	0.00e+00	5.00e-06	0.00e+00	0.00e+00
S365	1.72e-07	0.00e+00	2.00e-06	6.43e-04	6.98e-03	4.35e-17	0.00e+00	7.00e-06	1.70e-04	1.72e-02

Table 34: Feasibility error $\|\max\{\psi(x_k), 0\}\|_\infty$ for small scale noisy constrained `CUTEst` problems.

	$\tau = 10^{-6}$						$\tau = 10^{-2}$					
	KNITRO			COBYLA			KNITRO			COBYLA		
problem	$\phi(x)$	feaserr	#fevals	$\phi(x)$	feaserr	#fevals	$\phi(x)$	feaserr	#fevals	$\phi(x)$	feaserr	#fevals
CB3	2.000000	0.00e+00	35	2.000000	0.00e+00	30	1.9999980	6.00e-06	30	2.000030	1.20e-05	20
CHACONN2	1.999998	6.00e-06	30	2.000000	0.00e+00	29	1.9999980	6.00e-06	30	1.999631	5.01e-04	22
GIGOMEZ3	2.000000	0.00e+00	35	2.000000	0.00e+00	24	2.0000000	0.00e+00	35	2.000000	0.00e+00	24
HS100	680.630087	8.00e-06	173	680.630055	1.15e-04	125	680.7482460	0.00e+00	75	680.707367	0.00e+00	87
DIPIGRI	680.630025	3.10e-05	189	680.630055	2.80e-05	205	680.7513980	0.00e+00	75	680.638226	2.42e-04	114
HS93	135.080301	1.00e-06	650	135.092558	2.00e-06	f	135.0937190	1.00e-06	477	135.080622	5.37e-04	535
HS64	6299.795253	2.10e-05	77	6300.065790	1.20e-05	312	6688.2748180	5.70e-05	30	6704.980195	1.24e-04	238
POLAK3	5.933006	1.00e-06	357	5.933005	1.19e-04	1333	5.9332460	1.55e-04	268	5.939929	0.00e+00	1220
POLAK1	2.718281	1.00e-06	50	2.718279	9.00e-06	251	2.7182600	5.50e-05	45	2.743571	7.00e-05	216
HS104	3.951218	0.00e+00	305	3.952662	1.00e-06	f	3.9535870	1.20e-05	198	3.953450	4.63e-04	353
HS100MOD	678.679616	5.42e-04	146	678.679644	7.50e-05	179	678.7677270	4.00e-06	110	678.849653	0.00e+00	83
TWOBARS	1.508653	0.00e+00	55	1.508652	0.00e+00	40	1.5086550	3.00e-06	45	1.508930	0.00e+00	21
HS85	-2.215605	0.00e+00	152	-2.215605	9.00e-06	246	-2.2156220	5.41e-04	145	-2.206122	0.00e+00	233
CB2	1.952224	2.00e-06	57	1.952225	1.00e-06	48	1.9522210	4.00e-06	52	1.960966	0.00e+00	22
CHACONN1	1.952225	1.00e-06	40	1.952225	1.00e-06	37	1.9521680	8.20e-05	35	1.955056	0.00e+00	20
MADSEN	0.616433	0.00e+00	58	0.616432	4.74e-04	32	0.6164330	0.00e+00	58	0.616264	5.11e-04	31
HS66	0.518164	0.00e+00	33	0.518163	0.00e+00	37	0.5186740	0.00e+00	21	0.518026	3.71e-04	26
MINMAXBD	115.706388	1.17e-04	485	115.706134	4.03e-04	783	115.7063880	1.17e-04	485	115.919506	0.00e+00	741
CANTILVR	1.339957	0.00e+00	211	1.339958	3.80e-05	131	1.3415130	0.00e+00	187	1.347590	0.00e+00	107
HS92	1.374297	0.00e+00	f	1.367672	0.00e+00	139	1.3742970	0.00e+00	f	1.366663	2.00e-06	117
HALDMADS	0.000121	6.00e-06	78	0.000159	1.00e-06	f	0.0001210	6.00e-06	78	0.000159	1.00e-06	f
HS34	-0.834033	1.00e-06	54	-0.834032	0.00e+00	34	-0.8320120	0.00e+00	44	-0.834016	0.00e+00	25
HS90	1.362766	0.00e+00	326	1.368155	1.00e-06	f	1.3608930	2.00e-06	206	1.360847	1.10e-05	156
SNAKE	-22.242365	1.70e-03	f	-0.021173	2.00e-06	f	-22.2423650	1.70e-03	f	-0.021173	2.00e-06	f
HS72	727.747619	0.00e+00	f	727.695652	0.00e+00	870	721.1362080	3.33e-04	66	720.999346	3.14e-04	815
GIGOMEZ2	1.952224	1.00e-06	64	1.952225	1.80e-05	46	1.9520790	2.15e-04	59	1.952432	0.00e+00	31
SYNTHES1	0.759272	1.60e-05	64	0.759285	1.00e-06	53	0.7591430	1.95e-04	56	0.785862	1.00e-06	30
HS88	1.362528	0.00e+00	331	1.362528	0.00e+00	114	1.3581270	5.00e-06	240	1.354190	8.00e-06	93
HS13	0.989068	0.00e+00	97	1.006571	0.00e+00	f	1.1183300	0.00e+00	36	1.138030	0.00e+00	19
MATRIX2	0.000000	0.00e+00	153	0.000001	0.00e+00	113	0.0000000	0.00e+00	153	0.000001	0.00e+00	113
WOMFLET	0.000029	1.50e-05	f	0.000006	6.00e-05	121	0.0230460	4.21e-04	81	0.000468	3.02e-04	68
S365	0.000000	5.41e-04	54	0.000000	3.13e-04	48	0.0000000	5.41e-04	54	0.000000	3.13e-04	48

Table 35: Number of function evaluations to obtain (4.9) for noise level $\sigma_f = 10^{-7}$.

	$\tau = 10^{-6}$						$\tau = 10^{-2}$					
	KNITRO			COBYLA			KNITRO			COBYLA		
problem	$\phi(x)$	feaserr	#fevals	$\phi(x)$	feaserr	#fevals	$\phi(x)$	feaserr	#fevals	$\phi(x)$	feaserr	#fevals
CB3	2.000003	0.00e+00	47	2.000000	8.00e-06	32	1.999981	2.50e-05	30	2.000031	1.10e-05	20
CHACONN2	2.000006	4.00e-06	45	2.000001	2.79e-04	31	1.999999	1.90e-05	30	1.999624	5.08e-04	22
GIGOMEZ3	2.000003	0.00e+00	47	2.000003	1.26e-03	25	2.000003	0.00e+00	47	2.000003	1.26e-03	25
HS100	680.630284	9.00e-06	160	680.630275	1.47e-04	152	680.849165	0.00e+00	46	680.757842	1.82e-03	56
DIPIGRI	680.630221	7.00e-06	149	680.630243	5.00e-05	164	680.817892	1.54e-03	56	680.718994	3.36e-04	58
HS93	135.138021	1.00e-05	229	135.416906	1.60e-05	f	135.133971	1.16e-03	66	135.416906	1.60e-05	f
HS64	6299.871889	0.00e+00	66	6299.950263	2.61e-03	308	6708.036271	2.43e-03	25	6669.894009	2.09e-03	246
POLAK3	5.933216	8.00e-06	313	5.933217	7.40e-05	2303	5.932429	3.09e-03	270	5.943411	4.15e-03	2186
POLAK1	2.718282	6.00e-06	50	2.718280	1.20e-05	269	2.718182	9.00e-04	45	2.743588	4.30e-05	230
HS104	3.964210	0.00e+00	170	3.963888	1.04e-04	f	3.964996	3.71e-04	109	3.965500	9.61e-04	113
HS100MOD	678.679887	1.30e-05	193	678.679885	5.49e-04	122	678.947334	1.28e-03	73	678.953589	0.00e+00	52
TWOBARS	1.508641	9.00e-06	155	1.508665	1.90e-05	f	1.508463	1.28e-04	41	1.508969	0.00e+00	21
HS85	-2.215605	1.70e-05	106	-2.215605	3.30e-05	243	-2.215628	6.85e-04	99	-2.206170	8.10e-05	235
CB2	1.952220	1.40e-05	60	1.952219	7.00e-06	64	1.957493	6.23e-04	44	1.960991	0.00e+00	22
CHACONN1	1.952218	1.40e-05	58	1.952220	1.40e-05	42	1.952172	6.10e-05	35	1.955070	0.00e+00	20
MADSEN	0.616425	9.00e-06	146	0.616425	1.17e-04	40	0.616422	2.61e-04	49	0.613403	3.27e-03	20
HS66	0.518177	3.00e-06	f	0.518158	1.00e-05	37	0.518164	3.50e-05	32	0.518055	1.87e-03	19
MINMAXBD	115.706237	5.45e-04	926	115.705937	1.72e-03	779	115.705316	2.33e-03	919	122.679517	0.00e+00	719
CANTILVR	1.339958	8.00e-06	149	1.339959	3.30e-05	151	1.338856	2.67e-03	91	1.342286	4.59e-03	111
HS92	1.244708	2.58e-04	f	1.112153	6.73e-04	140	1.244708	2.58e-04	f	1.109888	7.50e-04	68
HALDMADS	0.000133	1.40e-05	101	0.000179	1.40e-05	f	0.000133	1.40e-05	101	0.000135	5.10e-05	77
HS34	-0.834035	6.00e-06	52	-0.834035	3.20e-05	32	-0.833901	0.00e+00	36	-0.834014	0.00e+00	25
HS90	2.061544	1.20e-05	f	1.049278	1.07e-03	118	2.061544	1.20e-05	f	1.049423	1.07e-03	72
SNAKE	-2.275095	2.12e-04	87	-0.032419	3.00e-06	f	-2.269176	2.34e-04	69	-0.032419	3.00e-06	f
HS72	702.418590	7.31e-04	f	528.392603	7.94e-03	f	702.418590	7.31e-04	f	528.392603	7.94e-03	f
GIGOMeZ2	1.952237	0.00e+00	81	1.952209	1.80e-05	f	1.952214	1.60e-03	49	1.952439	0.00e+00	31
SYNTHES1	0.759275	6.00e-06	122	0.759278	2.40e-05	113	0.758501	1.34e-04	56	0.839094	0.00e+00	29
HS88	1.324205	4.40e-05	f	1.180569	3.57e-04	67	1.324205	4.40e-05	f	1.177325	3.68e-04	47
HS13	0.954433	1.20e-05	331	1.029338	5.00e-06	f	1.089502	0.00e+00	36	1.138525	0.00e+00	19
MATRIX2	0.000012	9.00e-06	135	0.000038	6.00e-06	f	0.000012	9.00e-06	135	0.000038	6.00e-06	f
WOMFLET	0.001388	1.00e-06	f	-0.000003	2.33e-03	80	0.017147	1.63e-03	90	0.001884	1.73e-04	68
S365	0.000000	6.32e-04	54	0.000000	5.30e-03	36	0.000000	6.32e-04	54	0.000000	5.30e-03	36

Table 36: Number of function evaluations to obtain (4.9) for noise level $\sigma_f = 10^{-5}$

	$\tau = 10^{-6}$						$\tau = 10^{-2}$					
	KNITRO			COBYLA			KNITRO			COBYLA		
problem	$\phi(x)$	feaserr	#fevals	$\phi(x)$	feaserr	#fevals	$\phi(x)$	feaserr	#fevals	$\phi(x)$	feaserr	#fevals
CB3	2.000135	7.50e-05	57	2.001105	0.00e+00	f	1.991529	1.86e-02	25	1.999922	9.37e-04	17
CHACONN2	2.000382	1.20e-04	30	2.000360	1.84e-03	f	1.989816	2.59e-02	25	2.007665	3.71e-02	19
GIGOMEZ3	2.000135	7.50e-05	f	1.999908	3.32e-04	35	2.000135	7.50e-05	f	1.999908	3.32e-04	35
HS100	680.643454	6.18e-04	118	680.653293	2.80e-03	f	680.862278	2.56e-03	47	680.751163	1.19e-02	46
DIPIGRI	680.642602	5.53e-04	146	680.653238	2.48e-03	f	680.858601	0.00e+00	47	680.938844	1.37e-02	46
HS93	135.324499	1.03e-03	126	136.540776	7.65e-03	f	135.323447	3.39e-03	35	136.540776	7.65e-03	f
HS64	6294.941500	2.18e-03	74	6307.172876	1.82e-01	f	6746.496044	5.01e-02	20	6307.172876	1.82e-01	f
POLAK3	5.937398	1.34e-03	f	5.932593	8.61e-03	1645	5.954694	7.66e-03	252	5.909504	3.84e-02	1510
POLAK1	2.719811	2.51e-04	86	2.752280	9.16e-04	f	2.723341	7.78e-04	36	2.736065	3.33e-02	231
HS104	3.981084	1.04e-03	174	4.087923	2.80e-02	f	3.981888	9.32e-04	154	4.087923	2.80e-02	f
HS100MOD	678.700188	3.91e-04	f	678.688238	0.00e+00	99	678.964451	3.27e-03	73	678.817053	2.85e-02	61
TWOBARS	1.508513	1.28e-03	f	1.507091	1.04e-03	27	1.508513	1.28e-03	f	1.507577	1.34e-02	21
HS85	-2.209889	1.52e-04	271	-1.733123	1.41e-03	f	-2.211306	1.94e-02	127	-1.733123	1.41e-03	f
CB2	1.952771	0.00e+00	f	1.952216	1.02e-03	45	1.943877	1.91e-02	51	1.956972	0.00e+00	23
CHACONN1	1.956186	0.00e+00	f	1.952905	7.34e-04	23	1.949855	4.28e-02	46	1.934334	2.62e-02	15
MADSEN	0.616321	4.89e-04	f	0.616142	3.03e-04	35	0.619826	0.00e+00	68	0.612382	1.21e-02	26
HS66	0.518819	0.00e+00	f	0.517640	1.21e-03	48	0.518819	0.00e+00	f	0.518165	1.35e-02	13
MINMAXBD	115.710191	2.96e-04	f	115.706307	4.03e-02	756	115.703385	1.16e-02	538	121.478546	2.68e-02	717
CANTILVR	1.341675	0.00e+00	141	1.347345	3.04e-03	f	1.343901	5.05e-03	84	1.336533	4.03e-02	145
HS92	0.003678	1.33e-01	f	0.507997	2.58e-02	f	0.003678	1.33e-01	f	0.507997	2.58e-02	f
HALDMADS	0.007637	0.00e+00	f	0.000616	9.93e-04	71	0.007637	0.00e+00	f	0.000616	9.93e-04	71
HS34	-0.833608	0.00e+00	71	-0.834809	1.73e-03	f	-0.832197	0.00e+00	36	-0.832210	0.00e+00	22
HS90	0.001271	1.33e-01	f	0.683684	1.11e-02	f	0.001271	1.33e-01	f	0.683684	1.11e-02	f
SNAKE	-45.074608	2.90e-03	130	-0.035633	3.27e-04	f	-44.926682	3.22e-02	60	-0.035633	3.27e-04	f
HS72	349.372500	4.62e-02	f	66.439486	3.47e-01	f	349.372500	4.62e-02	f	66.439486	3.47e-01	f
GIGOMeZ2	1.951903	6.68e-04	f	1.951310	1.29e-03	48	1.951903	6.68e-04	f	1.951561	7.74e-04	36
SYNTHES1	0.766084	0.00e+00	76	0.755256	1.12e-03	f	0.790862	1.53e-03	56	0.844177	0.00e+00	29
HS88	0.958819	2.05e-03	f	0.575949	1.82e-02	45	0.958819	2.05e-03	f	0.576448	1.81e-02	26
HS13	0.857313	4.07e-04	48	1.114815	0.00e+00	f	0.857380	4.06e-04	38	1.114815	0.00e+00	f
MATRIX2	0.003633	7.30e-04	f	0.002521	3.17e-04	68	0.003633	7.30e-04	f	0.002521	3.17e-04	68
WOMFLET	0.008386	4.53e-04	f	0.003122	0.00e+00	108	0.007706	5.15e-02	69	-0.051536	5.38e-02	87
S365	0.000000	4.03e-02	45	0.000000	5.11e-02	32	0.000000	4.03e-02	45	0.000016	4.64e-02	25

Table 37: Number of function evaluations to obtain (4.9) for noise level $\sigma_f = 10^{-3}$.

	$\tau = 10^{-6}$						$\tau = 10^{-2}$					
	KNITRO			COBYLA			KNITRO			COBYLA		
problem	$\phi(x)$	feaserr	#fevals	$\phi(x)$	feaserr	#fevals	$\phi(x)$	feaserr	#fevals	$\phi(x)$	feaserr	#fevals
CB3	2.151366	0.00e+00	200	2.297095	3.71e-02	f	2.151799	0.00e+00	183	2.297095	3.71e-02	f
CHACONN2	1.917274	2.54e-01	f	0.544480	1.53e+00	f	1.917274	2.54e-01	f	0.544480	1.53e+00	f
GIGOMEZ3	2.162155	0.00e+00	f	1.558092	5.89e-01	f	2.162155	0.00e+00	f	1.554036	5.32e-01	20
HS100	682.785696	1.01e-01	f	680.545842	2.09e-01	f	682.785696	1.01e-01	f	680.679716	3.84e-01	25
DIPIGRI	682.612679	1.38e-01	f	681.544064	0.00e+00	49	682.612679	1.38e-01	f	681.282921	7.16e-02	39
HS93	132.388506	1.13e-01	126	135.053154	4.58e-02	f	132.388506	1.13e-01	126	135.053154	4.58e-02	f
HS64	7178.300921	4.96e-01	33	13921.561135	4.97e+00	f	6221.151364	5.05e-01	27	13921.561135	4.97e+00	f
POLAK3	5.768461	5.56e-01	f	-2.341289	8.72e+00	f	5.768461	5.56e-01	f	-2.341289	8.72e+00	f
POLAK1	5.342043	6.74e-02	86	7.870717	9.39e-02	f	5.381476	1.35e-01	62	7.870717	9.39e-02	f
HS104	3.287811	2.81e-01	f	1.208522	3.70e-01	30	3.287811	2.81e-01	f	1.193347	3.97e-01	10
HS100MOD	680.016339	1.77e-01	f	679.878287	7.18e-02	45	680.111489	8.61e-02	53	679.599540	2.77e-01	34
TWOBARS	1.601395	1.86e-02	f	1.450060	1.40e-01	30	1.601395	1.86e-02	f	1.450155	1.40e-01	12
HS85	-1.288983	0.00e+00	30	-1.256810	0.00e+00	f	-1.288983	0.00e+00	30	-1.256810	0.00e+00	f
CB2	2.157319	1.15e-02	f	0.871901	1.10e+00	f	2.157319	1.15e-02	f	0.871901	1.10e+00	f
CHACONN1	2.085730	2.43e-03	f	1.976579	2.38e-02	27	2.085730	2.43e-03	f	1.979692	2.45e-02	12
MADSEN	0.874476	0.00e+00	70	0.882487	0.00e+00	f	0.874516	0.00e+00	33	0.882487	0.00e+00	f
HS66	0.543480	0.00e+00	f	0.509050	4.56e-02	37	0.543480	0.00e+00	f	0.508822	4.63e-02	20
MINMAXBD	-33.388429	4.65e+02	f	112.975518	2.81e+00	f	-33.388429	4.65e+02	f	112.975518	2.81e+00	f
CANTILVR	1.531159	5.74e-02	f	0.934009	2.47e+00	f	1.531159	5.74e-02	f	0.934009	2.47e+00	f
HS92	0.366123	1.35e-01	f	0.293398	7.11e-02	48	0.366123	1.35e-01	f	0.297797	7.09e-02	35
HALDMADS	0.252628	0.00e+00	f	-3.027998	1.82e+01	f	0.252628	0.00e+00	f	-3.027998	1.82e+01	f
HS34	-0.141457	1.15e-01	49	-0.750941	2.99e-01	f	-0.140985	1.15e-01	25	-0.750941	2.99e-01	f
HS90	0.790027	1.40e-01	f	0.331080	1.20e-01	48	0.790027	1.40e-01	f	0.331906	1.20e-01	13
SNAKE	-8.350669	1.87e-01	93	1.284550	3.65e-01	f	-8.350669	1.87e-01	93	1.284550	3.65e-01	f
HS72	67.968573	5.76e-01	f	8.946566	3.26e+00	f	67.968573	5.76e-01	f	8.946566	3.26e+00	f
GIGOMeZ2	1.874401	1.78e-01	78	2.056211	0.00e+00	f	1.874484	1.90e-01	70	2.056211	0.00e+00	f
SYNTHES1	1.977343	2.09e-01	f	1.855036	0.00e+00	11	1.841763	3.65e-01	69	1.855036	0.00e+00	11
HS88	0.047086	1.26e-01	16	0.235414	1.47e-01	f	0.047086	1.26e-01	16	0.235414	1.47e-01	f
HS13	0.357703	6.52e-02	48	2.381364	0.00e+00	f	0.551282	1.71e-02	19	2.381364	0.00e+00	f
MATRIX2	0.075268	2.95e-02	194	1.186858	0.00e+00	f	0.075268	2.95e-02	194	1.186858	0.00e+00	f
WOMFLET	6.254269	5.05e-02	f	3.338088	0.00e+00	48	6.254269	5.05e-02	f	3.336520	0.00e+00	35
S365	0.000000	9.22e-02	45	0.000000	5.72e-02	24	0.000000	9.22e-02	45	-0.001477	1.53e-01	23

Table 38: Number of function evaluations to obtain (4.9) for noise level $\sigma_f = 10^{-1}$.