NORTHWESTERN UNIVERSITY

Methods for Nonlinear and Noisy Optimization

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Industrial Engineering and Management Science

By

Yuchen Xie

EVANSTON, ILLINOIS

September 2021

ABSTRACT

Methods for Nonlinear and Noisy Optimization

Yuchen Xie

In this thesis, we aim to develop efficient algorithms with theoretical guarantees for noisy nonlinear optimization problems, with and without constraints, under various different assumptions. Apart from Chapter 1 which provides relevant backgrounds, the remaining of thesis is divided into four chapters.

In Chapter 2, we establish the theoretical convergence results for a modified BFGS method when bounded noise is presented in both function and gradient evaluations. In order to guarantee the stability of BFGS update in the presence of such noise, we propose to modify the classic BFGS method by incorporating a procedure called *lengthening*, which spaces out the points at which gradient differences are collected when necessary. We show that on strongly convex functions, the proposed BFGS method is globally convergent to a neighborhood of the minimizer, whose size depends on the noise level.

The lengthening procedure described in Chapter 2 requires the knowledge of the strongly convex parameter of the objective function, making it difficult to apply the proposed method in realistic settings. In Chapter 3, we build upon the ideas in Chapter

2 and propose a practical noise-tolerant BFGS/limited-memory BFGS (L-BFGS) method which addresses this difficulty. We design a new line search that is capable of performing adaptive lengthening without knowing exogenous function information. We prove that the proposed method is globally convergent on strongly convex functions, and performs better than unmodified BFGS/L-BFGS on noisy optimization problems both in terms of efficiency and the accuracy it reaches.

In Chapter 4, we consider the derivative-free optimization (DFO) where we only have access to the noisy objective function but not its derivatives. One approach under this setting is computing an approximation of the derivative using finite-difference, and in order for this approach to be efficient, the finite-difference interval needs to be chosen appropriately. A typical choice is $\sqrt{\epsilon_M}$, but this choice can perform poorly when we have noise in function evaluations. To address this issue, we propose an estimation procedure that performs a bisection search on the finite-difference interval to find near-optimal difference interval. We show that under suitable assumptions, the proposed procedure terminates in finite iterations, and outputs optimal differencing intervals (up to constant). In addition, this procedure can be generalized to any finite difference scheme beyond forward difference and central difference, including schemes for estimating higher-order derivatives. We also demonstrate its reliability and accuracy on a number of noisy functions.

Apart from bounded noise we consider in previous chapters, another frequently seen type of noise is stochastic noise. In Chapter 5, we consider a constrained stochastic optimization problem, where the objective function is computed in expectation, and the variables are subject to deterministic and simple convex constraints. A common method

for such problems is the mini-batching stochastic proximal gradient method, which computes a stochastic gradient for the objective function using a mini-batch of samples, and applies the proximal gradient method. One difficulty in this method is choosing the size of mini-batch appropriately so that the overall sample complexity is kept to a minimum. To achieve this goal, we describe an adaptive sampling strategy, which adaptively increases the batch size as the iterates approach the minimizer. We present global convergence results for the proposed algorithm, on both strongly convex and general convex objective functions. Numerical experiments are also presented to demonstrate the efficacy of the proposed method.

Acknowledgements

It is only with the unwavering support of many people that I can achieve this milestone in my life, and I would like to express my sincerest gratitude to them.

First and foremost, I would like to thank my advisor Professor Jorge Nocedal. It is a great honor for me to be advised by and work alongside one of the great minds in optimization. Thank you for being a great advisor who has patiently guided me, once an outsider to the field of optimization, into such a beautiful world. Thank you for being a great mentor and offering me so much advice, guidance, and opportunities for my academic and career development. Thank you for being a great role model to whom I shall always look up. Thank you for being a great friend, who is so caring, understanding, supporting, encouraging, and always willing to help. I am incredibly fortunate to have you throughout this journey.

I would like to thank many mentors I have had throughout my Ph.D. studies, all of whom I have learned a great deal from. Thank you, Professor Andreas Wächter and Professor Ermin Wei, for teaching me so much and serving on my dissertation committee. Thank you, Professor Richard Byrd, for the opportunities to collaborate with you and for your help with many challenging problems I've encountered. Thank you, Professor Zhaoran Wang, for leading me into the world of machine learning. I have also had the privilege of collaborating with many other exceptional researchers: Raghu, Michael, Yi, and Melody. Thank you all for your efforts in our research projects.

I want to thank the IEMS faculty for offering me admission to the Ph.D. program five years ago, teaching me so much about industrial engineering, and being role models as exceptional researchers. I also want to thank the IEMS staff members for always being so warm and helpful, especially Stephen and Agnes.

I also want to thank my lab mates who have shared office L375 with me: Albert, Francisco, Raghu, Alejandra, Ruby, Michael, Melody, Oliver, Xinyi, Shigeng, and Shima. Thank you for warmly welcoming me into the optimization lab, for discussing stimulating ideas with me, for putting up with me when I am noisy, and for making me feel like a part of a family.

The past five years have been incredibly happy, for which I have my dear friends to thank. Ruby and Yi, when I began my Ph.D. studies five years ago, I could never have imagined that I could make such dear friends like you, with whom I would share so many memories. We've had so many ups and downs, went past so many milestones in our journey, and we went through all of them together. Although we are no longer in the same city, I believe that our friendship will be long-lasting. Yi'an, you're always so passionate, cheerful while also intelligent and insightful. It was a pleasure to have you as a dear friend, a role model, and a roommate. Muchen, thank you for all the fun we've had and for all the help you've given me when I need it. Yintai, Liwei, Xin, and so many more friends, thank you all for bringing joy to my life. I could not list all the names here, but the memories with these friends will always be in my heart.

Last but not least, I want to thank my family members. Mum and Dad, although you are not here with me during my Ph.D. studies, your unconditional love and support are what keep me forward in this journey. I want to thank my grandma, grandpa, and elder

sister for always believing in and caring about me. Sisi, my fiancée, thank you for always believing in me, for seeing the best in me, and for encouraging me when I'm doubting myself. In the first two years of my Ph.D. life, we lived in different countries and could only see each other once in several months. Thank you so much for changing your career path entirely, going across the ocean to come to this country and be with me. I love you and thank you for being in my life.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

Optimization problems occur in a wide range of fields including natural science, social science and engineering, and have been the fundamental tool people use for decision-making. Most of the optimization algorithms have been developed under the implicit assumption that the objective functions, including their derivatives, can be evaluated exactly. In practice however, more often than not the objective functions are evaluated approximately and contain some noise. Such noise may be a result of the computation process through which the objective function is evaluated, for example, round-off errors from floating-point arithmetics. Another source of noise is stochastic optimization, where objective function is expressed as the expectation of a random variable and is usually approximated by a sample average. Such settings are very common in simulation optimization and machine learning.

The optimization problem considered in this thesis can be generally expressed as:

$$\min_{x \in \mathcal{C}} \phi(x),$$

where $\mathcal{C} \subseteq \mathbb{R}^n$ is the feasible set, and can be either $\mathbb{R}^n$ itself (unconstrained optimization) or a subset of $\mathbb{R}^n$ (constrained optimization). However, we must perform this minimization only by observing *noisy* function evaluations

$$f(x) = \phi(x) + \epsilon(x),$$

and in some cases, noisy gradient evaluations

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

where $\epsilon(x)$ and $e(x)$ model the noise in function and gradient evaluations, respectively. Depending on the specific applications, $\epsilon(x)$ and $e(x)$ can either be deterministic functions, or random variables that follow specific distributions.

In Chapter 2 and 3, we consider applying the BFGS and the limited-memory BFGS (L-BFGS) method to an unconstrained, noisy optimization problem, with access to noisy function and gradient evaluations where the noise is assumed to be uniformly bounded:

$$|\epsilon(x)| \leq \epsilon_f, \ \|e(x)\| \leq \epsilon_g.$$

In Chapter 4, we consider the scenario where we only have access to the noisy function but *not* the gradient, a setting commonly known as derivative-free optimization (DFO), with the same uniform boundedness assumption. In Chapter 5, we consider a constrained, stochastic optimization problem, where $\epsilon(x)$ and $e(x)$ are zero-mean random variables. Each chapter is self-contained, and may use its own set of notations.

In the remaining parts of this chapter, we briefly discuss the backgrounds, introducing relevant optimization problems and algorithms, and motivating our investigations.

1.1. Quasi-Newton Methods for Noisy Optimization Problems

Quasi-Newton methods are a class of optimization methods for unconstrained optimization problem:

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

where $f(\cdot)$ is continuously differentiable. They utilize the displacement $s_k = x_{k+1} - x_k$ and the change in the gradients $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ between consecutive iterations to extract second-order information of the objective function $f(\cdot)$, and maintain the fast local convergence speed of Newton's method without having to evaluate the Hessian, or Hessian-vector products.

1.1.1. BFGS Method

The Broyden-Fletcher-Goldfarb-Shanno (BFGS) method with Armijo-Wolfe line search (Algorithm 1.1), is among the most successful and widely-used quasi-Newton methods. Starting from a positive definite matrix $H_0 \succ 0$, it iteratively updates matrix H_k as the approximation of the inverse of the Hessian $[\nabla^2 f(x_k)]^{-1}$ by (1.1), which is equivalent to solving the following constrained optimization problem:

$$\begin{aligned} H_{k+1} = \ & \underset{H}{\operatorname{argmin}} \ \|H - H_k\|_W^2, \ Ws_k = y_k \\ & \text{s.t. } Hy_k = s_k, \quad H^T = H, \end{aligned}$$

where $\|\cdot\|_W$ represents the weighted Frobenius norm and W is *any* positive definite matrix such that $Ws_k = y_k$.

Line search is a procedure in which the step size α_k is determined adaptively, a key ingredient of the BFGS method. The most common line search used in conjunction with BFGS method is the Armijo-Wolfe line search, which attempts to find a step size that

satisfies the Armijo-Wolfe condition:

$$\phi(x_k + \alpha_k p_k) \leq \phi(x_k) + c_1 \alpha p_k^T \nabla\phi(x_k) \qquad \text{(Armijo condition)}$$

$$p_k^T \nabla\phi(x_k + \alpha_k p_k) \geq c_2 p_k^T \nabla\phi(x_k), \qquad \text{(Wolfe condition)}$$

The first condition is also called the **sufficient decrease condition** because it requires the function to decrease sufficiently at the new iterate, generally preventing the step size from being too large; and the second condition is also called the **curvature condition**, which requires the directional derivative of the objective function along the search direction p_k to increase sufficiently, generally preventing the step size from being too small.

It is well known [**48**] that the Armijo-Wolfe line search can stabilize the BFGS updates. Moreover, for any function with Lipschitz continuous gradients that is bounded from below, the standard bi-sectioning Armijo-Wolfe line search terminates finitely, generating a step size that satisfies the Armijo-Wolfe condition.

The BFGS method with Armijo-Wolfe line search (Algorithm 1.1) has many nice properties: it is well-defined for any objective function $\phi(\cdot)$ that is continuously differentiable and bounded from below [**48**]; globally convergent if $\phi(\cdot)$ is convex, twice continuously differentiable with bounded level sets [**51**]; and locally superlinear convergent if $\phi(\cdot)$ is also strongly convex [**18**]. While examples of non-convex functions on which the method fails to converge exist [**24, 41**], the method performs very well in practice.

Algorithm 1.1: *BFGS Method with Armijo-Wolfe Line search*

Input: smooth function $\phi(\cdot)$; constants $0 < c_1 < c_2 < 1$; starting point x_0; initial Hessian inverse approximation $H_0 \succ 0$.

1: **for** $k = 0, 1, 2, ...,$ **do**

2: $p_k \leftarrow -H_k \nabla \phi(x_k)$

3: **Line search**: find a stepsize α_k that satisfies **the Armijo-Wolfe condition**:

$$\phi(x_k + \alpha_k p_k) \leq \phi(x_k) + c_1 \alpha_k p_k^T \nabla \phi(x_k)$$
$$p_k^T \nabla \phi(x_k + \alpha_k p_k) \geq c_2 p_k^T \nabla \phi(x_k)$$

4: Take the step:

$$x_{k+1} \leftarrow x_k + \alpha_k p_k$$

5: Compute the *curvature pair* (s_k, y_k):

$$s_k = x_{k+1} - x_k, \quad y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

6: Update inverse Hessian approximation using the curvature pairs (s_k, y_k):

$$H_{k+1} = \left(I - \rho_k s_k y_k^T\right) H_k \left(I - \rho_k y_k s_k^T\right) + \rho_k s_k s_k^T, \text{ where } \rho_k = \frac{1}{s_k^T y_k} \tag{1.1}$$

7: **end for**

1.1.2. Limited-Memory BFGS Method

The limited-memory BFGS (L-BFGS) method is a modification of the BFGS method, which aims to reduce the computational and storage cost of the BFGS method on large-scale optimization problems. It is among the state-of-the-art algorithms for large-scale unconstrained optimization problems.

On a n-dimensional problem, the BFGS method has a per-iteration computational cost of $O(n^2)$ and a storage cost of $O(n^2)$. For modern large-scale optimization problems, an $O(n^2)$ cost might be prohibitively high, making it impractical to apply the BFGS method.

Instead of maintaining a full $n \times n$ matrix H_k which encodes all the information of previous curvature pairs $\{(s_j, y_j), j = 0, 1, \cdots, k-1\}$, the L-BFGS method only keeps a limited number, m (usually $m \ll n$), of *most recent* curvature pairs, reducing the storage cost from $O(n^2)$ to $O(mn)$. The inverse Hessian approximation matrix H_k used in L-BFGS method is constructed as follows: starting from $H_k^0 \succ 0$, which is allowed to vary from iteration to iteration, iteratively apply BFGS update (1.1) using curvature pairs (s_{k-m}, y_{k-m}), (s_{k-m+1}, y_{k-m+1}), $\cdots$ (s_{k-1}, y_{k-1}) to obtain a series of matrices $H_k^1, H_k^2, \cdots, H_k^m$, and let $H_k = H_k^m$.

In L-BFGS method, the matrix H_k is implicitly defined by these m most recent curvature pairs and is never constructed explicitly. Instead, L-BFGS method relies on the *two loop recursion* to compute the product $H_k \nabla \phi_k$, with a computational cost of only $O(mn)$ if H_k^0 is chosen appropriately [**48**, Chapter 7.2, page 178].

Just like the BFGS method, the L-BFGS method also uses the Armijo-Wolfe line search to compute the step size α_k, where the line search plays a similar role of stablizing the updates. The L-BFGS method with Armijo-Wolfe line search is shown to be R-linearly convergent on strongly convex functions [**40**]; unlike the BFGS method, the L-BFGS method does not converge superlinearly. Interestingly, all existing analyses of the L-BFGS indicate a convergence rate that is much slower than the gradient descent method, which is in sharp contrast to the efficacy of the L-BFGS method in practice. Developing a

non-pessemistic analysis for the L-BFGS method that matches its performance in practice remains an open problem.

1.1.3. BFGS and L-BFGS Method on Noisy Functions

Both the BFGS method and L-BFGS method are developed under the implicit assumption that the function $\phi(x)$ and gradient $\nabla\phi(x)$ evaluations are exact. If we only have access to noisy evaluations of function and gradient, i.e., if we only have access to $f(x)$ and $g(x)$ where

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

then both methods will face two challenges.

First of all, the Armijo-Wolfe line search is no longer guaranteed to success. Without the presence of noise, one can show that there exists intervals of stepsizes that satisfy the Armijo-Wolfe condition so long as p_k is a descent direction for ϕ at x_k, and that $\phi(x_k + \alpha p_k)$ is bounded from below for $\alpha \geq 0$ [**48**, Chapter 3.1, page 35, Lemma 3.1]. In the presence of noise, the Armijo-Wolfe condition must be replaced by:

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k p_k^T g(x_k)$$

$$p_k^T g(x_k + \alpha_k p_k) \geq c_2 p_k^T g(x_k)$$

The existence of a stepsize α_k such that the above conditions are satisfied is no longer guaranteed. This means the line search procedure could fail and the algorithm is no longer well-defined.

Another more subtle issue is the contamination of the BFGS matrix. Since BFGS matrix H_k is updated iteratively using (1.1), it retains information in the curvature pair (s_k, y_k). In the presence of noise, y_k in the curvature pair is "contaminated" by the noise in the gradient, as evident in the following decomposition:

$$y_k = g(x_k + \alpha_k p_k) - g(x_k) = \underbrace{(\nabla\phi(x_k + \alpha_k p_k) - \nabla\phi(x_k))}_{\text{true gradient difference}} + \underbrace{(e(x_k + \alpha_k p_k) - e(x_k))}_{\text{"noise"}}$$

Even if the Wolfe condition can still guarantee $s_k^T y_k > 0$ (assuming the line search succeeds), using this contaminated information can result in deterioration of the BFGS matrix H_k, resulting in stagnation of the algorithm. As an example, we plot the condition number of H_k generated by the vanilla BFGS method on the `ARWHEAD` problem from the CUTEst problem set [**30**], with independent, uniformly distributed noise added to each component of the gradient; see Figure 1.1. We can see that the condition number of H_k can grow to 10^{15}, which signifies its deterioration due to the contamination of noise. We will revisit this example in Chapter 3. Because of the limited-memory of H_k, this issue is mitigated for the L-BFGS method, but it can still adversely affect the optimization progress.

In Chapter 2 and 3, we will address these two challenges by modifying the vanilla BFGS and L-BFGS method. We will establish the conditions under which the line search procedure is guaranteed to succeed, and address the issue of contamination of H_k through a procedure we call "lengthening".

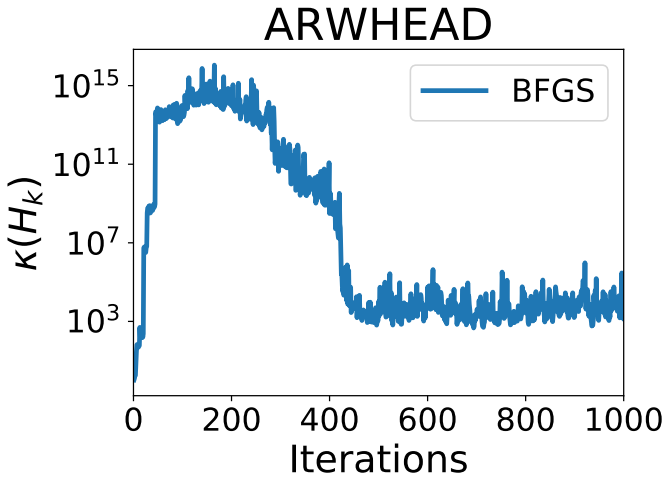

Figure 1.1. Condition number of H_k, generated by BFGS method on a simple noisy function (`ARWHEAD` from CUTEst problem set).

1.2. Derivative-free Optimization and Finite Difference

In many applications, we only have access to the value of the objective function, but not its derivative. This setting is commonly known as the derivative-free optimization (DFO), or black-box optimization.

The optimization problem in the DFO setting can be expressed as

$$\min_{x\in\mathbb{R}^n} \; f(x), \text{ where } f(x) = \phi(x) + \epsilon(x)$$

where $\phi(x)$ is a smooth function and $\epsilon(x)$ is the noise in the function.

There are a plethora of algorithms for DFO problem; see, for example, [**37**] for a thorough review of existing optimization algorithms for DFO. In this thesis, we will focus on the finite-difference-based approaches, which approximate the gradient by using the finite-difference approximation.

Finite-difference is a simple yet effective method to compute an approximation of the gradient when it is not available. The basic idea of finite-difference is using the Taylor expansion. Since function $\phi(x)$ is smooth, for any $s \in \mathbb{R}^n$ and $h \in \mathbb{R}$, we have:

$$\phi(x + h \cdot s) = \phi(x) + h \cdot s^T \nabla\phi(x) + O(h^2).$$

Therefore, we can approximate a component of $\nabla\phi(x)$ by:

$$[\nabla\phi(x)]_i \approx \frac{\phi(x + h \cdot e_i) - \phi(x)}{h},$$

where e_i is the unit vector along i-th coordinate. In practice however, we only have access to $f(x)$ instead of $\phi(x)$, and therefore we use:

$$[\nabla\phi(x)]_i \approx \frac{f(x + h \cdot e_i) - f(x)}{h}. \tag{1.2}$$

This approach is known as *forward difference*, and $h > 0$ is called *differencing interval*. Alternatively, one can also use:

$$[\nabla\phi(x)]_i \approx \frac{f(x + h \cdot e_i) - f(x - h \cdot e_i)}{2h}; \tag{1.3}$$

this approach is known as *central difference*.

If we assume $\phi(x)$ is sufficiently smooth, by Taylor's theorem, we have:

$$\phi(x + h \cdot s) = \phi(x) + h \cdot s^T \nabla\phi(x) + \frac{h^2}{2} \cdot s^T \nabla^2\phi(x)s + O(h^3),$$

and we can see the error in the forward difference approximation is bounded by:

$$\left|\frac{f(x+h\cdot e_i)-f(x)}{h}-e_i^T\nabla\phi(x)\right| \leq \underbrace{\frac{h}{2}\cdot\left|e_i^T\nabla^2\phi(x)e_i\right|+O(h^2)}_{\text{error due to truncation}}+\underbrace{\frac{2\epsilon_f}{h}}_{\text{error due to noise}} \tag{1.4}$$

From (1.4) we can see that the error in forward difference approximation comes from two sources: error due to truncation, i.e., due to higher order terms in the Taylor expansion, and error due to noise in $f(x)$. The choice of differencing interval h can affect the accuracy of forward difference approximation: decreasing h will decrease the error due to truncation, but will increase the error due to noise in f. Therefore, there exists an optimal choice of h which balances these two sources of error. If we ignore $O(h^2)$ term in (1.4), the optimal choice of h is:

$$h_i^* = 2\sqrt{\frac{\epsilon_f}{|e_i^T\nabla^2\phi(x)e_i|}}.$$

For central differencing scheme (1.3), a similar analysis exists, and the optimal choice of h depends on the bound on the third order derivative of ϕ.

Unfortunately, such choices require the knowledge of both noise level ϵ_f and high-order derivative of ϕ. While the noise level can be estimated by using procedures like `ECNoise` [**43**], estimating high-order derivative is very difficult in the DFO setting.

In Chapter 4, we address this difficulty by desgining a bisectioning procedure that search for the correct differencing interval which balances the error due to truncation and due to noise, without any knowledge of the high-order derivatives. The procedure applies to both forward differencing and central differencing scheme, and can be generalized to arbitrary finite differencing schemes.

1.3. Stochastic Optimization and Adaptive Sampling

In many applications, including supervised machine learning and simulation optimization, it is of interest to solve the following optimization problem:

$$\min_{x\in\mathbb{R}^n} \; F(x) \overset{\text{def}}{=\!=} \mathbb{E}_\xi \left[f(x;\xi) \right], \; \xi \sim \mathcal{D},$$

where ξ is a random variable following certain distribution $\mathcal{D}$, and $f(x;\xi)$ is a smooth function of x for all possible values of ξ. A very popular method for solving this problem is the mini-batching stochastic gradient method, which approximate the gradient of $F(x)$ via sample average approximation:

$$\nabla F(x) \approx g_m(x) \overset{\text{def}}{=\!=} \frac{1}{m}\sum_{j=1}^{m} \nabla_x f(x;\xi_j), \text{ where } \xi_1, \xi_2, \cdots, \xi_m \overset{\text{iid}}{\sim} \mathcal{D}.$$

The parameter m is called *batch size*, and it controls the quality of the gradient approximation. For a gradient approximateion computed with a batch size of m, the error in the approximation is of order $O(1/\sqrt{m})$.

While increasing batch size improves the quality of gradient approximation, it also increases computational and storage cost. It is therefore desirable to balance this trade-off by adaptively controlling batch size: in the beginning of the optimization, a crude approximation is sufficient to make progress, and a small m should be used; as the iterates approach the minimizer, however, a more refined approximation is needed and m must be increased. A natural idea is to control the error in $g_m(x)$ such that it is comparable to $\|\nabla F(x)\|$, i.e., to choose m such that

$$\mathbb{E}\left[\|g_m(x) - \nabla F(x)\|^2 \right] \leq \theta^2 \|\nabla F(x)\|^2$$

for some $\theta \in (0, 1)$. This is known as the *norm test* [**16**]. It can be shown [**16**] that under some assumptions, the idealized version of the norm test can ensure linear convergence to the minimizer. In practice, the exact gradient $\nabla F(x)$ is not available and can only be estimated.

Nevertheless, the idea behind the norm test cannot directly generalize to the case of *constrained optimization*. Consider the constrained optimization setting:

$$\min_{x \in \Omega} \; F(x) \stackrel{\text{def}}{=\!=} \mathbb{E}_{\xi} \left[f(x; \xi) \right], \; \xi \sim \mathcal{D},$$

where $\Omega \subseteq \mathbb{R}^n$ is the feasible set. The reason is that for constrained optimization, the norm of gradient $\|\nabla F(x)\|$ is no longer a valid measure of progress. Unlike the unconstrained case, $\|\nabla F(x)\|$ does not converge to 0 even if x converges to a stationary point. In Chapter 5, we investigate this setting and propose a natural generalization of the norm test so that it applies to constrained optimization.

CHAPTER 2

Analysis of the BFGS Method with Errors

2.1. Introduction

The behavior of the BFGS method in the presence of errors has received little attention in the literature. There is, however, an increasing interest in understanding its theoretical properties and practical performance when functions and gradients are inaccurate. This interest is driven by applications where the objective function contains noise, as is the case in machine learning, and in applications where the function evaluation is a simulation subject to computational errors. The goal of this chapter is to extend the theory of quasi-Newton methods to the case when there are errors in the function and gradient evaluations. We analyze the classical BFGS method with a slight modification consisting of lengthening the differencing interval as needed; all other aspects of the algorithm, including the line search, are unchanged. We establish global convergence properties on strongly convex functions. Specifically, we show that if the errors in the function and gradient are bounded, the iterates converge to a neighborhood of the solution whose size depends on the level of noise (or error).

Our analysis builds upon the results in [**18**], which identify some fundamental properties of BFGS updating. The extension to the case of inaccurate gradients is not simple due to the complex nature of the quasi-Newton iteration, where the step affects the Hessian update, and vice versa, and where the line search plays an essential role. The existing

analysis relies on the observation that changes in gradients provide reliable curvature estimates, and on the fact that the line search makes decisions based on the true objective function. In the presence of errors, gradient differences can give misleading information and result in poor quasi-Newton updates. Performance can further be impaired by the confusing effects of a line search based on inaccurate function information. We show that these difficulties can be overcome by our modified BFGS algorithm, which performs efficiently until it reaches a neighborhood of the solution where progress is no longer possible due to errors.

The proposed algorithm aims to be a natural adaptation of the BFGS method that is capable of dealing with noise. Other ways of achieving robustness might include update skipping and modifications of the curvature vectors, such as Powell damping [**52**]. We view these as less desirable alternatives for reasons discussed in the next section. The line search strategy could also be performed in other ways. For example, in their analysis of a gradient method with errors, Berahas et al. [**4**], relax the Armijo condition to take noise into account. We prefer to retain the standard Armijo-Wolfe line search without any modification, as this has practical advantages.

The literature of the BFGS method with inaccurate gradients includes the implicit filtering method of Kelley et al. [**22, 36**], which assumes that noise can be diminished at will at any iteration. Deterministic convergence guarantees have been established for that method by ensuring that noise decays as the iterates approach the solution. Dennis and Walker [**26**] and Ypma [**59**] study bounded deterioration properties and local convergence of quasi-Newton methods with errors, when started near the solution with a Hessian approximation that is close to the exact Hessian. Barton [**2**] proposes an implementation

of the BFGS method in which gradients are computed by an appropriate finite differencing technique, assuming that the noise level in the function evaluation is known. Berahas et al. [**4**] estimate the noise in the function using Hamming's finite difference technique [**34**], as extended by Moré and Wild [**43**], and employ this estimate to compute a finite difference gradient in the BFGS method. They analyze a gradient method with a relaxation of the Armijo condition, but do not study the effects of noise in BFGS updating.

There has recently been some interest in designing quasi-Newton methods for machine learning applications using stochastic approximations to the gradient [**17, 32, 45, 56**]. These papers avoid potential difficulties with BFGS or L-BFGS updating by assuming that the quality of gradient differences is always controlled, and as a result, the analysis follows similar lines as for classical BFGS and L-BFGS.

This chapter is organized in 5 sections. The proposed algorithm is described in Section 2.2. Section 2.3, the bulk of the chapter, presents a sequence of lemmas related to the existence of stepsizes that satisfy the Armijo-Wolfe conditions, the beneficial effect of lengthening the differencing interval, the properties of "good iterates", culminating in a global convergence result. Some numerical tests that illustrate the performance of the method with errors in the objective function and gradient are given in Section 2.4. The chapter concludes in Section 2.5 with some final remarks.

2.2. The Algorithm

We are interested in solving the problem

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

where the function $\phi \in C^1$ and its gradient $\nabla\phi$ are not directly accessible. Instead, we have access to inaccurate (or noisy) versions, which we denote as $f(x)$ and $g(x)$, respectively. Thus, we write

$$\begin{aligned} f(x) &= \phi(x) + \epsilon(x) \\ g(x) &= \nabla\phi(x) + e(x), \end{aligned} \tag{2.1}$$

where $\epsilon(x)$ and $e(x)$ define the error in function and gradient values. [1] To apply the BFGS method, or a modification of it, to minimize the true function ϕ, while observing only noisy function and gradient estimates, we must give careful consideration to the two main building blocks of the BFGS method: the line search and Hessian updating procedures.

As was shown by Powell [**50**], an Armijo-Wolfe line search guarantees the stability of the BFGS updating procedure, and ultimately the global convergence of the iteration (for convex objectives). In the deterministic case, when the smooth function $\phi(x)$ and its gradient are available, this line search computes a stepsize α that satisfies:

$$\begin{aligned} \phi(x + \alpha p) &\leq \phi(x) + c_1 \alpha p^T \nabla\phi(x) \qquad \text{(Armijo condition)} \\ p^T \nabla\phi(x + \alpha p) &\geq c_2 p^T \nabla\phi(x), \qquad \text{(Wolfe condition)} \end{aligned} \tag{2.2}$$

where x is the current iterate, p is a descent direction for ϕ at x, (i.e., $p^T\nabla\phi(x) < 0$), and $0 < c_1 < c_2 < 1$ are user-specified parameters. The first condition imposes sufficient decrease in the objective function, and the second requires an increase in the directional derivative (and is sometimes referred to as the *curvature condition*). It is well known [**48**]

[1] We express $f(x)$ and $g(x)$ as single valued functions, but they could validly be viewed as arbitrary members of a set valued function at x, where all members of the set satisfy an error bound.

that if $\phi \in C^1$ is bounded below and has Lipschitz continuous gradients, there exists an interval of steplengths α that satisfy (2.2).

When $\phi(x)$ and $\nabla\phi(x)$ are not accessible, it is natural to attempt to satisfy the Armijo-Wolfe conditions for the noisy function and gradient, i.e., to find $\alpha > 0$ such that

$$\begin{aligned} f(x+\alpha p) &\leq f(x) + c_1 \alpha p^T g(x) \\ p^T g(x+\alpha p) &\geq c_2 p^T g(x), \end{aligned} \tag{2.3}$$

where p is the BFGS search direction. It is, however, not immediately clear whether such a stepsize exists, and if it does, whether it satisfies the Armijo-Wolfe conditions (2.2) for the true function ϕ.

One possible approach to address these two challenges is to relax the Armijo-Wolfe conditions (2.3), as is done e.g. by Berahas et al. [**4**] in their analysis of a gradient method with errors. An alternative, which we adopt in this chapter, is to keep the Armijo-Wolfe conditions unchanged, and show that under suitable conditions there is a stepsize that satisfies the Armijo-Wolfe conditions for both the noisy and true objective functions. Our main assumption is that the errors $\epsilon(x), e(x)$ in (2.1) are bounded for all x.

Let us now consider the BFGS updating procedure. The key in the convergence analysis of quasi-Newton methods is to show that the search direction is not orthogonal to the gradient. In the literature on Newton-type methods, this is usually done by bounding the condition number of the Hessian approximation B_k. Whereas this is possible for limited memory quasi-Newton methods, such as L-BFGS, in which B_k is obtained by performing a limited number of updates, one cannot bound the condition number of B_k for the standard BFGS method without first proving that the iterates converge to the

solution. Nevertheless, there is a result about BFGS updating [**18**], for strongly convex objective functions, whose generality will be crucial in our analysis. It states that for a fixed *fraction* of the BFGS iterates, the angle between the search direction and the gradient is bounded away from 90°.

To apply the results in [**18**], we need to ensure that the update of B_k is performed using correction pairs

$$[s_k, y_k] = [(x_{k+1} - x_k), (g(x_{k+1}) - g(x_k))]$$

that satisfy,

$$\frac{y_k^T s_k}{s_k^T s_k} \geq \widehat{m}, \qquad \frac{y_k^T y_k}{y_k^T s_k} \leq \widehat{M}, \quad \forall k, \tag{2.4}$$

for some constants $0 < \widehat{m} \leq \widehat{M}$. The Armijo-Wolfe line search does not, however, guarantee that these conditions are satisfied in our setting, even under the assumption that ϕ is strongly convex. To see this, note that when $\|s_k\|$ is small compared to the gradient error, the vector y_k can be contaminated by errors, and (2.4) may not hold. In other words, difficulties arise when the differencing interval is too short, and to overcome this problem, we modify the ordinary BFGS method by lengthening the differencing interval, as needed. How to do this will be discussed in the next section.

Based on these observations, we provide in Algorithm 2.1 a description of the proposed method. In what follows, we let H_k denote the inverse Hessian approximation; i.e, $H_k = B_k^{-1}$. We say the line search *succeeded* if it satisfies the Armijo-Wolfe condition after a limited number of function evaluations. This issue is discussed further in section 3.4.

Algorithm 2.1: *Outline of the BFGS Method with Errors*

Input: functions $f(\cdot)$ and $g(\cdot)$; constants $0 < c_1 < c_2 < 1$; lengthening parameter $xl > 0$; starting point x_0; initial Hessian inverse approximation $H_0 \succ 0$.

for $k = 0, 1, 2, ...,$ **do**

  $p_k \leftarrow -H_k g(x_k)$

  Attempt to find a stepsize α^* such that

$$f(x_k + \alpha^* p_k) \leq f(x_k) + c_1 \alpha^* p_k^T g(x_k)$$

$$p_k^T g(x_k + \alpha^* p_k) \geq c_2 p_k^T g(x_k)$$

  if Succeeded **then**

    $\alpha_k \leftarrow \alpha^*$

  else

    $\alpha_k \leftarrow 0$

  end if

  if $\|\alpha_k p_k\| \geq l$ **then**

    Compute the curvature pair as usual:

$$s_k \leftarrow \alpha_k p_k, \; y_k \leftarrow g\left(x_k + s_k\right) - g(x_k)$$

  else

    Compute the curvature pair by lengthening the search direction:

$$s_k \leftarrow l \frac{p_k}{\|p_k\|}, \quad y_k \leftarrow g\left(x_k + s_k\right) - g(x_k)$$

  end if

  Update the inverse Hessian approximation using the curvature pairs (s_k, y_k):

$$H_{k+1} = \left(I - \rho_k s_k y_k^T\right) H_k \left(I - \rho_k y_k s_k^T\right) + \rho_k s_k s_k^T, \text{ where } \rho_k = \frac{1}{s_k^T y_k} \tag{2.5}$$

  $x_{k+1} \leftarrow x_k + \alpha_k p_k$

end for

Note that step 12 in Algorithm 2.1 requires an additional gradient evaluation. The only unspecified parameter in this algorithm is the lengthening parameter l, whose choice will be studied in the next section. We note for now that l needs only be large enough to compensate for the error in the gradient, and should be at least of order $O(\epsilon_g)$, where ϵ_g is a bound on the gradient error. Even though step 12 is executed when the line search fails, we will show below that the lengthening operation guarantees that $s_k^T y_k > 0$ so that the BFGS update is well defined.

As mentioned in Section 2.1, lengthening the step is not the only way to stabilize the BFGS update in the presence of errors. One alternative is to skip the update, but this can prevent the algorithm from building a useful Hessian approximation. One can also modify the curvature vector y_k when the stability of the BFGS updating cannot be guaranteed, but it is difficult to know how to design this modification in the presence of noise in the function and gradient. We choose the lengthening approach because we view it as well suited in the presence of noise.

2.3. Convergence Analysis

In this section, we give conditions under which the BFGS method outlined above is guaranteed to yield an acceptable solution, by which we mean a function value that is within the level of noise of the problem. Throughout the chapter, $\|\cdot\|$ denotes the ℓ_2 norm.

Our analysis relies on the following assumptions regarding the true objective function ϕ and the errors in function and gradients.

Assumption 2.3.1. *The function $\phi(x)$ is bounded below and is twice continuously differentiable with an M-Lipschitz continuous ($M > 0$) gradient, i.e.,*

$$\|\nabla\phi(x) - \nabla\phi(y)\| \leq M \|x - y\|, \ \forall x, y \in \mathbb{R}^d.$$

This assumption could be relaxed to require only that the gradients be Lipschitz continuous; we make the stronger assumption that $\phi \in C^2$ only to simplify the proof of one of the lemmas below.

Assumption 2.3.2. *The errors in function and gradients values are uniformly bounded i.e., $\forall x \in \mathbb{R}^d$, there exist non-negative constants ϵ_f, ϵ_g such that*

$$|f(x) - \phi(x)| = |\epsilon(x)| \leq \epsilon_f$$

$$\|g(x) - \nabla\phi(x)\| = \|e(x)\| \leq \epsilon_g.$$

There are many applications where this assumption holds; one of the most prominent is the case of computational noise that arises when the evaluation of the objective function involves an adaptive numerical computation [**43**]. On the other hand, there are other applications where Assumption 2.3.2 is not satisfied, as is the case when errors are due to Gaussian noise. Nevertheless, since the analysis for unbounded errors appears to be complex [**23**], we will not consider it here, as our main goal is to advance our understanding of the BFGS method in the presence of errors, and this is best done, at first, in a benign setting.

2.3.1. Existence of Armijo-Wolfe Stepsizes

We begin our analysis by presenting a result that will help us establish the existence of stepsizes satisfying the Armijo-Wolfe conditions. Since we will impose these conditions on the noisy functions (i.e. (2.3)), we want to show that the Armijo-Wolfe conditions also hold for the true function. The following lemma considers two sets of functions and gradients: F_A and G_A can be viewed as proxies for the true function and gradient ϕ and $\nabla\phi$, while F_B and G_B stand for the approximate function f and its gradient approximation g. (In a later lemma these roles are reversed.) It is intuitively clear, that the Armijo-Wolfe conditions can only be meaningful when the gradients are not dominated by errors. Therefore, our first lemma shows that when the gradients G_A, G_B are sufficiently large compared to ϵ_f, ϵ_g, the Armijo-Wolfe conditions can be satisfied.

Below, we let φ denote the angle between a vector $p \in \mathbb{R}^d$ and a vector $-G \in \mathbb{R}^d$, i.e.,

$$\varphi = \angle(p, -G) \quad \text{or} \quad \cos\varphi = \frac{-p^T G}{\|p\|\|G\|}. \tag{2.6}$$

In the following lemma, φ_A, φ_B denote the angles obtained by substituting G_A, G_B in this definition.

Lemma 2.3.1. *Suppose that a scalar function $F_A : \mathbb{R}^d \to \mathbb{R}$ is continuous and bounded below, and that a vector function $G_A : \mathbb{R}^d \to \mathbb{R}^d$ satisfies*

$$\|G_A(y) - G_A(z)\| \leq L\,\|y - z\| + \Lambda, \quad \forall y, z \in \mathbb{R}^d, \tag{2.7}$$

for some constants $L > 0$, and $\Lambda \geq 0$. Suppose $x \in \mathbb{R}^d$ is such that $G_A(x) \neq 0$, that $p \in \mathbb{R}^d$ satisfies $p^T G_A(x) < 0$, and that the stepsize $\alpha > 0$ satisfies the Armijo-Wolfe

conditions

$$\begin{aligned} F_A(x+\alpha p) &\leq F_A(x) + c_{A1}\alpha p^T G_A(x) \\ p^T G_A(x+\alpha p) &\geq c_{A2} p^T G_A(x), \end{aligned} \tag{2.8}$$

for $0 < c_{A1} < c_{A2} < 1$. *Furthermore, consider another scalar function* $F_B : \mathbb{R}^d \to \mathbb{R}$ *and vector function* $G_B : \mathbb{R}^d \to \mathbb{R}^d$ *satisfying*

$$\begin{aligned} |F_A(y) - F_B(y)| &\leq \epsilon_f, \ \forall y \in \mathbb{R}^d \\ \|G_A(y) - G_B(y)\| &\leq \epsilon_g, \ \forall y \in \mathbb{R}^d, \end{aligned} \tag{2.9}$$

for some non-negative constants ϵ_f, ϵ_g. *For the same* x *and* p, *suppose that* $G_B(x) \neq 0$ *and that* p *satisfies* $p^T G_B(x) < 0$. *Let* γ_1, γ_2 *be two constants such that*

$$0 < \gamma_1 < c_{A1} \quad \textit{and} \quad 0 < \gamma_2 < 1 - c_{A2}. \tag{2.10}$$

If the following conditions hold:

$$\begin{aligned} \|G_A(x)\| &\geq \frac{2\Lambda}{(1-c_{A2})\cos\varphi_A} \\ \|G_B(x)\| &\geq \max\left\{ \frac{2c_{A1}\epsilon_g}{\gamma_1 \cos\varphi_B}, \ \frac{(1+c_{A2})\epsilon_g}{\gamma_2 \cos\varphi_B} \right\} \\ \|G_A(x)\| \, \|G_B(x)\| &\geq \frac{8L\epsilon_f}{\gamma_1(1-c_{A2})\cos\varphi_A \cos\varphi_B}, \end{aligned} \tag{2.11}$$

then the stepsize α *satisfies the Armijo-Wolfe conditions with respect to* F_B *and* G_B*:*

$$F_B(x+\alpha p) \leq F_B(x) + (c_{A1} - \gamma_1)\alpha p^T G_B(x) \tag{2.12}$$

$$p^T G_B(x+\alpha p) \geq (c_{A2} + \gamma_2) p^T G_B(x). \tag{2.13}$$

Proof. By the second equation in (2.8), i.e.,

$$p^T G_A(x+\alpha p) \geq c_{A2} p^T G_A(x),$$

we have

$$-(1-c_{A2}) p^T G_A(x) \leq p^T \big(G_A(x+\alpha p) - G_A(x)\big).$$

Using (2.7) we have

$$-(1-c_{A2}) p^T G_A(x) \leq \|p\| \left(\alpha L \|p\| + \Lambda\right).$$

Recalling the definition (2.6), we obtain the lower bound

$$\alpha \geq \frac{(1-c_{A2}) \cos\varphi_A \|G_A(x)\| - \Lambda}{L\|p\|}.$$

From (2.11) we have

$$\|G_A(x)\| \geq \frac{2\Lambda}{(1-c_{A2})\cos\varphi_A},$$

i.e.,

$$(1-c_{A2}) \cos\varphi_A \|G_A(x)\| \geq 2\Lambda,$$

from which it follows that

$$\alpha \geq \underline{\alpha} \stackrel{\text{def}}{=} \frac{(1-c_{A2}) \cos\varphi_A \|G_A(x)\|}{2L\|p\|}.$$

Now, by (2.11) we also have

$$\|G_A(x)\| \|G_B(x)\| \geq \frac{8L\epsilon_f}{\gamma_1 (1-c_{A2}) \cos\varphi_A \cos\varphi_B},$$

and thus

$$\begin{aligned} -\gamma_1 \alpha p^T G_B(x) &\geq -\gamma_1 \underline{\alpha} p^T G_B(x) \\ &= \gamma_1 \frac{(1-c_{A2})\cos\varphi_A \|G_A(x)\|}{2L\|p\|} \|p\| \|G_B(x)\| \cos\varphi_B \\ &= \frac{\gamma_1(1-c_{A2})\cos\varphi_A \cos\varphi_B}{2L} \|G_A(x)\| \|G_B(x)\| \geq 4\epsilon_f. \end{aligned} \tag{2.14}$$

From (2.11)

$$\|G_B(x)\| \geq \frac{2c_{A1}\epsilon_g}{\gamma_1 \cos\varphi_B},$$

or equivalently

$$-\gamma_1 \alpha p^T G_B(x) \geq 2c_{A1}\alpha \|p\| \epsilon_g. \tag{2.15}$$

Adding (2.14) and (2.15) yields

$$-\gamma_1 \alpha p^T G_B(x) \geq 2\epsilon_f + c_{A1}\alpha \|p\| \epsilon_g. \tag{2.16}$$

The first inequality in (2.8) and (2.9) give

$$F_B(x+\alpha p) \leq F_B(x) + c_{A1}\alpha p^T G_B(x) + 2\epsilon_f + c_{A1}\alpha \|p\| \epsilon_g,$$

which combined with (2.16) yields

$$F_B(x+\alpha p) \leq F_B(x) + (c_{A1}-\gamma_1)\alpha p^T G_B(x). \tag{2.17}$$

This proves (2.12).

Next, by (2.11)

$$\|G_B(x)\| \geq \frac{(1+c_{A2})\epsilon_g}{\gamma_2 \cos\varphi_B},$$

or equivalently

$$-(1+c_{A2})\epsilon_g \|p\| \geq \gamma_2 p^T G_B(x). \tag{2.18}$$

By the second equation in (2.8) and (2.9) we immediately have

$$p^T G_B(x+\alpha p) \geq c_{A2} p^T G_B(x) - (1+c_{A2})\epsilon_g \|p\| .$$

Then by (2.18) we have

$$p^T G_B(x+\alpha p) \geq (c_{A2}+\gamma_2) p^T G_B(x),$$

which proves (2.13). □

Note that there is some flexibility in the choice of γ_1, γ_2 in (2.10), which influences the constants in (2.11). This lemma gives conditions under which the Armijo-Wolfe conditions hold, but the bounds (2.11), involve the angles φ_A, φ_B, which have not been shown to be bounded away from 90° (so that the cosine terms are not bounded away from zero). Hence, this result is preliminary. We continue the analysis leaving the angles φ_A, φ_B as parameters to be bounded later.

In the sequel we let $g_k = g(x_k)$, and for nonzero $g_k, p_k, \nabla\phi(x_k)$, we define θ_k to be the angle between p_k and $-g_k$, and we define $\widetilde{\theta}_k$ to be the angle between p_k and $-\nabla\phi(x_k)$. In

other words,

$$\theta_k = \angle(p_k, -g_k) \quad \text{or} \quad \cos(\theta_k) = -\, p_k^T g_k / \|p_k\| \|g_k\|, \tag{2.19}$$

$$\widetilde{\theta}_k = \angle(p_k, -\nabla\phi(x_k)) \quad \text{or} \quad \cos(\widetilde{\theta}_k) = -\, p_k^T \nabla\phi(x_k) / \|p_k\| \|\nabla\phi(x_k)\|. \tag{2.20}$$

We now use Lemma 2.3.1 to establish the existence of Armijo-Wolfe stepsizes for the noisy function and gradient, f and g, under the assumption that the true gradient $\nabla\phi$ is not too small.

Theorem 2.3.1. *Suppose that Assumptions 2.3.1 and 2.3.2 hold, and that at iteration k the search direction p_k satisfies $p_k^T g_k < 0$. Let $0 < c_1 < c_2 < 1$ and $0 < \delta_1 < 1$, $0 < \delta_2 < 1$ be constants such that $\delta_1 + \delta_2 < c_2 - c_1$. If*

$$\|\nabla\phi(x_k)\| \geq \max\left\{ \frac{4(c_1+\delta_1)\epsilon_g}{\delta_1 \cos\theta_k}, \; \frac{2(1+c_2-\delta_2)\epsilon_g}{\delta_2 \cos\theta_k}, \right. \\ \left. \sqrt{\frac{16 M \epsilon_f}{(1-c_2+\delta_2)\delta_1 \cos\theta_k \cos\widetilde{\theta}_k}} \right\}, \tag{2.21}$$

there exists a stepsize α_k such that

$$\begin{aligned} f(x_k + \alpha_k p_k) &\leq f(x_k) + c_1 \alpha_k p_k^T g_k \\ p_k^T g(x_k + \alpha_k p_k) &\geq c_2 p_k^T g_k. \end{aligned} \tag{2.22}$$

Proof. We invoke Lemma 2.3.1 with $x \leftarrow x_k$, $F_A(\cdot) \leftarrow \phi(\cdot)$, $G_A(\cdot) \leftarrow \nabla\phi(\cdot)$, $F_B(\cdot) \leftarrow f(\cdot)$, $G_B(\cdot) \leftarrow g(\cdot)$, and $p \leftarrow p_k$. Then, from (2.19)-(2.20) we have that $\varphi_A = \widetilde{\theta}_k$ and $\varphi_B = \theta_k$. Let $\gamma_1 = \delta_1, \gamma_2 = \delta_2$; $c_{A1} = c_1 + \delta_1$ and $c_{A2} = c_2 - \delta_2$. Our assumptions on $\delta_1, \delta_2, c_1, c_2$ imply that $0 < c_{A1} < c_{A2} < 1$, and that conditions (2.10) hold.

We must verify that the assumptions of Lemma 2.3.1 are satisfied. By Assumption 2.3.1, F_A is bounded below and

$$\|G_A(y) - G_A(z)\| \leq M \|y - z\|,$$

so that (2.7) holds with $L = M$ and $\Lambda = 0$. We assume that $p^T G_B(x) = p_k^T g_k < 0$. To show that $p^T G_A(x) < 0$, note that by (2.21)

$$\|\nabla\phi(x_k)\| \geq \frac{4(c_1 + \delta_1)}{\delta_1} \frac{\epsilon_g}{\cos\theta_k} > 2\epsilon_g.$$

By Assumption 2.3.2, we have that $\|\nabla\phi(x_k) - g_k\| \leq \epsilon_g$. Therefore,

$$\|g_k\| \geq \|\nabla\phi(x_k)\| - \epsilon_g \geq \frac{1}{2} \|\nabla\phi(x_k)\|. \tag{2.23}$$

We also have that

$$\|g_k\| \geq \frac{1}{2} \|\nabla\phi(x_k)\| \geq \frac{2(c_1 + \delta_1)\epsilon_g}{\delta_1 \cos\theta_k} > \frac{\epsilon_g}{\cos\theta_k},$$

or

$$\|g_k\| \cos\theta_k > \epsilon_g.$$

Recalling again Assumption 2.3.2, this bound yields

$$\begin{aligned} {p_k}^T G_A(x) &\leq {p_k}^T G_B(x) + \|p_k\| \, \epsilon_g \\ &= - \|p_k\| \, (\|G_B(x)\| \cos \varphi_B - \epsilon_g) \\ &= - \|p_k\| \, (\|g_k\| \cos \theta_k - \epsilon_g) \\ &< 0. \end{aligned}$$

Since p_k is a descent direction for the true function ϕ, and since ϕ is continuously differentiable and bounded from below, we can guarantee [**48**] the existence of a stepsize $\alpha = \alpha_k$ such that

$$F_A(x + \alpha p) \leq F_A(x) + c_{A1} \alpha p^T G_A(x)$$

$$p^T G_A(x + \alpha p) \geq c_{A2} p^T G_A(x),$$

showing that (2.8) is satisfied.

To prove that (2.22) holds, all that is necessary is to show that (2.21) implies conditions (2.11). The first condition is immediately satisfied, since we have shown that we can choose $\Lambda = 0$. By the definitions given in the first paragraph of this proof, the other two conditions in (2.11) can be written as

$$\begin{aligned} \|g_k\| &\geq \max \left\{ \frac{2(c_1 + \delta_1)}{\delta_1}, \ \frac{(1 + c_2 - \delta_2)}{\delta_2} \right\} \frac{\epsilon_g}{\cos \theta_k} \\ \|\nabla \phi(x_k)\| \, \|g_k\| &\geq \frac{8 M \epsilon_f}{(1 - c_2 + \delta_2) \delta_1 \cos \theta_k \cos \widetilde{\theta}_k}. \end{aligned} \tag{2.24}$$

To see that these two conditions hold, we first note that by (2.23),

$$\|g_k\| \geq \frac{1}{2} \|\nabla \phi(x_k)\| \geq \max \left\{ \frac{2(c_1 + \delta_1)}{\delta_1} \frac{\epsilon_g}{\cos \theta_k}, \ \frac{(1 + c_2 - \delta_2)}{\delta_2} \frac{\epsilon_g}{\cos \theta_k} \right\}.$$

Also, from (2.21)

$$\|\nabla\phi(x_k)\|\,\|g_k\| \geq \frac{1}{2}\,\|\nabla\phi(x_k)\|^2 \geq \frac{8M\epsilon_f}{(1-c_2+\delta_2)\delta_1\cos\theta_k\cos\widetilde{\theta}_k}\,.$$

Hence, all the conditions of Lemma 2.3.1 are satisfied, and we conclude that there exists a stepsize α_k that satisfies (2.22). □

In the previous theorem we gave conditions under which the Armijo-Wolfe conditions are satisfied with respect to f and g. We now use Lemma 2.3.1 to show that satisfaction of the Armijo-Wolfe conditions for f implies satisfaction for the true objective ϕ, for the same steplength α_k, if $\|\nabla\phi(x_k)\|$ is sufficiently large.

Theorem 2.3.2. *Suppose Assumptions 2.3.1 and 2.3.2 are satisfied, and that at iteration k the search direction p_k satisfies $p_k^T g_k < 0$. Let θ_k and $\widetilde{\theta}_k$ be defined by* (2.19) *and* (2.20)*. Let $0 < c_1 < c_2 < 1$, and $\hat{\delta}_1, \hat{\delta}_2$ be constants such that $0 < \hat{\delta}_1 < c_1$, $0 < \hat{\delta}_2 < 1 - c_2$. Suppose there exists a stepsize α_k such that*

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1\alpha_k p_k^T g_k$$

$$p_k^T g(x_k + \alpha_k p_k) \geq c_2 p_k^T g_k.$$

If

$$\|\nabla\phi(x_k)\| \geq \max\left\{\frac{8\epsilon_g}{(1-c_2)\cos\theta_k},\ \sqrt{\frac{16M\epsilon_f}{\hat{\delta}_1(1-c_2)\cos\theta_k\cos\widetilde{\theta}_k}},\right.$$
$$\left.\frac{2c_1\epsilon_g}{\hat{\delta}_1\cos\widetilde{\theta}_k},\ \frac{(1+c_2)\epsilon_g}{\hat{\delta}_2\cos\widetilde{\theta}_k}\right\}, \tag{2.25}$$

then α_k satisfies

$$\begin{aligned} \phi(x_k + \alpha_k p_k) &\leq \phi(x_k) + (c_1 - \hat{\delta}_1)\alpha_k p_k^T \nabla\phi(x_k) \\ p_k^T \nabla\phi(x_k + \alpha_k p_k) &\geq (c_2 + \hat{\delta}_2) p_k^T \nabla\phi(x_k). \end{aligned} \tag{2.26}$$

Proof. We prove this by applying Lemma 2.3.1, reversing the roles of F_A, F_B, compared to Lemma 2.3.1. Specifically, we now let $x \leftarrow x_k$, $F_A(\cdot) \leftarrow f(\cdot)$, $G_A(\cdot) \leftarrow g(\cdot)$, $F_B(\cdot) \leftarrow \phi(\cdot)$, $G_B(\cdot) \leftarrow \nabla\phi(\cdot)$, and $p \leftarrow p_k$. We define $\varphi_A = \theta_k$ and $\varphi_B = \widetilde{\theta}_k$ as in (2.19), (2.20). Let $c_{A1} = c_1$, $c_{A2} = c_2$; $\gamma_1 = \hat{\delta}_1, \gamma_2 = \hat{\delta}_2$. Clearly we have $0 < c_{A1} < c_{A2} < 1$.

We need to verify that the assumptions of Lemma 2.3.1 are satisfied. By Assumptions 2.3.1 and 2.3.2 we have

$$\|G_A(y) - G_A(z)\| = \|g(y) - g(z)\| \leq \|\nabla\phi(y) - \nabla\phi(z)\| + 2\epsilon_g \leq M \|y - z\| + 2\epsilon_g,$$

and hence Assumption (2.7) is satisfied with $L = M$ and $\Lambda = 2\epsilon_g$.

We assume that $p^T G_A(x) = p^T g_k < 0$. To show that $p^T G_B(x) < 0$, we note from (2.25) that

$$\|\nabla\phi(x_k)\| \geq \frac{8\epsilon_g}{(1 - c_2)\cos\theta_k} > 2\epsilon_g,$$

and as in (2.23)

$$\|g_k\| \geq \|\nabla\phi(x_k)\| - \epsilon_g \geq \frac{1}{2}\|\nabla\phi(x_k)\|.$$

Therefore,

$$\|g_k\| \geq \frac{1}{2}\|\nabla\phi(x_k)\| \geq \frac{4\epsilon_g}{(1 - c_2)\cos\theta_k} > \frac{\epsilon_g}{\cos\theta_k}, \tag{2.27}$$

i.e,

$$\|g_k\| \cos\theta_k > \epsilon_g.$$

Now,

$$\begin{aligned} p^T G_B(x) &\leq p^T G_A(x) + \|p\| \, \epsilon_g \\ &= - \|p\| \, (\|G_A(x)\| \cos\varphi_A - \epsilon_g) \\ &= - \|p_k\| \, (\|g_k\| \cos\theta_k - \epsilon_g) \\ &< 0. \end{aligned}$$

It remains to show that conditions (2.11) are satisfied, from which it would follow that α_k satisfies (2.26), proving the theorem. Since $\Lambda = 2\epsilon_g$, conditions (2.11) read, in the notation of this lemma,

$$\begin{aligned} \|g_k\| &\geq \frac{4\epsilon_g}{(1-c_2)\cos\theta_k} \\ \|\nabla\phi(x_k)\| &\geq \max\left\{\frac{2c_1}{\hat{\delta}_1}, \ \frac{(1+c_2)}{\hat{\delta}_2}\right\} \frac{\epsilon_g}{\cos\widetilde{\theta}_k} \\ \|\nabla\phi(x_k)\| \, \|g_k\| &\geq \frac{8M\epsilon_f}{\hat{\delta}_1(1-c_2)\cos\theta_k \cos\widetilde{\theta}_k}. \end{aligned} \tag{2.28}$$

We have already shown, in (2.27), the first condition, and the second condition follows from (2.25). Finally, from (2.27) and (2.25),

$$\|\nabla\phi(x_k)\| \, \|g_k\| \geq \frac{1}{2} \|\nabla\phi(x_k)\|^2 \geq \frac{8M\epsilon_f}{\hat{\delta}_1(1-c_2)\cos\theta_k \cos\widetilde{\theta}_k}.$$

□

Theorems 2.3.1 and 2.3.2 establish the existence of a neighborhood of the solution, defined in terms of $\|\nabla\phi(x)\|$, outside of which the Armijo-Wolfe line search strategy is well defined. This neighborhood depends on ϵ_f and ϵ_g, as well as $\cos\theta_k$ and $\cos\widetilde{\theta}_k$ — and the latter two quantities have not yet been bounded away from zero. Thus, similar to the central role that $\cos\widetilde{\theta}_k$ plays in the classic convergence analysis of gradient methods, $\cos\theta_k$ and $\cos\widetilde{\theta}_k$ play a key role in the convergence analysis of our algorithm. The next step is to find a way to combine the results obtained so far with the theory developed in [**18**].

2.3.2. Lengthening the Differencing Interval

The BFGS method is complex in that Hessian updates affect the search direction and vice versa. As a result, it is not possible to show that the condition number of the Hessian approximations B_k is bounded without first showing convergence of the iterates. Nevertheless, it is has been shown [**18**] that under mild assumptions, the angle between the search direction and the negative gradient can be bounded away from zero for a fraction of the iterates, which is sufficient to establish R-linear convergence.

To apply the results in [**18**], the curvature pairs (s_k, y_k) used to update H_k must satisfy

$$\frac{y_k^T s_k}{s_k^T s_k} \geq \widehat{m}, \qquad \frac{y_k^T y_k}{y_k^T s_k} \leq \widehat{M}, \qquad \forall k, \tag{2.29}$$

for some constants $0 < \widehat{m} \leq \widehat{M}$. These conditions will not generally hold unless we make the following additional assumption.

Assumption 2.3.3. *The function ϕ is m-strongly convex, with $0 < m \leq M$. (Recall that M is defined in Assumption 2.3.1.)*

Assumptions 2.3.1, 2.3.2 and 2.3.3 are still not sufficient to establish (2.29) because, if $\|s_k\|$ is small compared to the error in the gradient, ϵ_g, then the vector y_k can be highly unreliable. In this case, we increase the differencing interval and recompute the gradient before performing the BFGS update, as stipulated in Algorithm 2.1, i.e., we set

$$s_k \leftarrow l\frac{p_k}{\|p_k\|}, \qquad y_k \leftarrow g\left(x_k + s_k\right) - g(x_k), \quad l > 0.$$

We show below that if the lengthening parameter l is sufficiently large, (2.29) holds. Lemma 2.3.3 identifies the minimum value of l. Before presenting that result, we need the following technical lemma. In what follows, $\lambda(H)$ denotes the set of eigenvalues of a matrix H.

Lemma 2.3.2. *Let $s, y \in \mathbb{R}^d$ be two non-zero vectors, and let $0 < \mu \leq L$. There exists a positive definite matrix $H \in \mathbb{S}^{d\times d}$ with eigenvalues $\lambda(H) \subseteq [\mu, L]$ such that*

$$y = Hs$$

if and only if

$$\left\| y - \frac{L+\mu}{2}s \right\| \leq \frac{L-\mu}{2}\|s\|. \tag{2.30}$$

Proof. Part I. We first show that if that $y = Hs$ with $\lambda(H) \subseteq [\mu, L]$ then (2.30) holds. Clearly,

$$\lambda\left(H - \frac{L+\mu}{2}I\right) \subseteq \left[-\frac{L-\mu}{2}, \frac{L-\mu}{2}\right].$$

Since $H - (L+\mu)I/2$ is symmetric, we have

$$\left\| H - \frac{L+\mu}{2} I \right\| \leq \frac{L-\mu}{2}.$$

Since

$$y - \frac{L+\mu}{2} s = \left(H - \frac{L+\mu}{2} I \right) s,$$

we conclude that

$$\left\| y - \frac{L+\mu}{2} s \right\| = \left\| \left(H - \frac{L+\mu}{2} I \right) s \right\| \leq \frac{L-\mu}{2} \|s\|.$$

Part II. We prove the converse by construction. To this end, we make the following *claim*: if $u, v \in \mathbb{R}^d$ are such that $\|u\| = \|v\| = 1$, then there exists a symmetric real matrix Q such that $Qu = v$ and $\lambda(Q) \subseteq \{-1, 1\}$. To prove this, we first note that if $u = -v$ then we can choose $Q = -I$. Otherwise, we have $u + v \neq 0$, and we let

$$e = \frac{u+v}{\|u+v\|}.$$

Then, a simple calculation shows that

$$Q = 2ee^T - I \tag{2.31}$$

satisfies $Qu = v$ and $Q^T = Q$. Since $\lambda(2ee^T) = \{0, 2\}$, we have $\lambda(Q) = \{-1, 1\}$, showing that our claim is true.

Now, to prove Part II, we assume that (2.30) holds. If

$$y - \frac{L+\mu}{2} s = 0,$$

then it follows immediately that $y = Hs$ with $\lambda(H) \subseteq [\mu, L]$. Otherwise, define

$$v = \frac{y - \frac{L+\mu}{2}s}{\left\|y - \frac{L+\mu}{2}s\right\|} \quad \text{and} \quad u = \frac{s}{\|s\|}.$$

We have shown above that since v, u are unit vectors, there exists a symmetric real matrix $Q \in \mathbb{S}_{d\times d}$ such that $v = Qu$ and $\lambda(Q) \subseteq \{-1, 1\}$, i.e.,

$$Q\frac{s}{\|s\|} = \frac{y - \frac{L+\mu}{2}s}{\left\|y - \frac{L+\mu}{2}s\right\|}.$$

Hence, we have

$$y = Hs, \text{ where } H = \frac{L+\mu}{2}I + \frac{\left\|y - \frac{L+\mu}{2}s\right\|}{\|s\|}Q.$$

Since we assume that

$$\frac{\left\|y - \frac{L+\mu}{2}s\right\|}{\|s\|} \leq \frac{L-\mu}{2},$$

and $\lambda(Q) \subseteq \{-1, 1\}$, we conclude that

$$\lambda(H) \subseteq [\mu, L].$$

□

With this result in hand, it is easy to establish the following bounds.

Lemma 2.3.3. *(Choice of the Lengthening Parameter) Suppose Assumptions 2.3.1, 2.3.2 and 2.3.3 hold. Let $s \in \mathbb{R}^d$ be a vector such that $\|s\| \geq l$, and define $y = g(x+s) - g(x)$. If*

$$l > 2\epsilon_g/m,$$

then

$$\begin{aligned} \frac{y^T s}{s^T s} &\geq \left(m - \frac{2\epsilon_g}{l}\right) \stackrel{\text{def}}{=\!=} \widehat{m} > 0 \\ \frac{y^T y}{y^T s} &\leq \left(M + \frac{2\epsilon_g}{l}\right) \stackrel{\text{def}}{=\!=} \widehat{M}. \end{aligned} \tag{2.32}$$

Proof. Let $\widetilde{y} = \nabla\phi(x+s) - \nabla\phi(x)$. Since $\phi \in C^2$, we have that $\widetilde{y} = As$, where A is the average Hessian

$$A = \int_0^1 \nabla^2\phi(x + t \cdot s)\, dt.$$

Since ϕ is m-strongly convex with M-Lipschitz continuous gradients, we know that $\lambda(A) \subseteq [m, M]$, and by Lemma 2.3.2 we have

$$\left\| \widetilde{y} - \frac{M+m}{2} s \right\| \leq \frac{M-m}{2} \|s\|. \tag{2.33}$$

By (2.1) and Assumption 2.3.2, we have

$$\|y - \widetilde{y}\| \leq 2\epsilon_g,$$

and hence

$$\left\| y - \frac{M+m}{2} s \right\| \leq \frac{M-m}{2} \|s\| + 2\epsilon_g.$$

If $\|s\| \geq l$, we have

$$\frac{M-m}{2} \|s\| + 2\epsilon_g \leq \frac{M-m}{2} \|s\| + \frac{2\epsilon_g}{l} \|s\|,$$

and thus

$$\left\| y - \frac{M+m}{2} s \right\| \leq \left(\frac{M-m}{2} + \frac{2\epsilon_g}{l}\right) \|s\|.$$

By defining

$$\widehat{m} = m - \frac{2\epsilon_g}{l}, \qquad \widehat{M} = M + \frac{2\epsilon_g}{l}, \tag{2.34}$$

we have

$$\left\| y - \frac{\widehat{M} + \widehat{m}}{2} s \right\| \leq \frac{\widehat{M} - \widehat{m}}{2} \|s\| .$$

Note that since $l > 2\epsilon_g/m$, we have $0 < \widehat{m} \leq \widehat{M}$. By Lemma 2.3.2, we know that there exists a positive definite matrix H with $\lambda(H) \subseteq [\widehat{m}, \widehat{M}]$ such that

$$y = Hs.$$

Then it immediately follows that

$$\frac{y^T s}{s^T s} \geq \widehat{m}, \qquad \frac{y^T y}{y^T s} \leq \widehat{M},$$

which proves the result due to (2.34). □

Therefore, if the lengthening parameter satisfies $l > 2\epsilon_g/m$, conditions (2.32) hold, as needed for the analysis that follows. This bound for l requires knowledge of the strong convexity parameter m, which may not be available in practice. One can show, however, that if we choose, at each iteration, $l > c\epsilon_g s^T s/y^T s$ where $c > 2$, the conditions (2.29) will be satisfied with different constants $\widehat{m}, \widehat{M}$. Although practical and simple to implement, this latter bound may not be the best in practice, as it results in a much larger $\widehat{M}$. We defer discussion of a practical determination of l to Chapter 3.

2.3.3. Properties of the "Good Iterates"

We now show that the angle between the search direction of Algorithm 2.1 and the true gradient is bounded away from 90°, for a fraction of all iterates. We begin by stating a result from [**18**, Theorem 2.1], which describes a fundamental property of the standard BFGS method (without errors).

Lemma 2.3.4. *(Existence of good iterates for classical BFGS) Let $H_0 \succ 0$, and let $\{H_k = B_k^{-1}\}$ be generated by the BFGS update (2.5) using **any** correction pairs $\{(s_k, y_k)\}$ satisfying (2.29) for all k. Define Θ_k to be the angle between s_k and $B_k s_k$, i.e.,*

$$\cos\Theta_k = \frac{s_k^T B_k s_k}{\|s_k\| \, \|B_k s_k\|}. \tag{2.35}$$

For a fixed scalar $q \in (0,1)$, let

$$\begin{aligned} \beta_0(q) &= \frac{1}{1-q}\left[\operatorname{tr}(B_0) - \log\det(B_0) + \hat{M} - 1 - \log\hat{m}\right] > 0 \\ \beta_1(q) &= e^{-\beta_0(q)/2} \in (0,1). \end{aligned} \tag{2.36}$$

Then we have, for all k,

$$\left|\left\{j \in \{0,1,\cdots,k-1\} \,\middle|\, \cos\Theta_j \geq \beta_1(q)\right\}\right| \geq qk. \tag{2.37}$$

We now establish a lower bound for the cosine of the angle between the quasi-Newton direction of Algorithm 2.1 and $-g_k$, i.e., a bound on $\cos\theta_k$ defined by setting $p_k \leftarrow -H_k g_k$ in (2.19).

Corollary 2.3.1. *Consider Algorithm 2.1 with lengthening parameter $l > 2\epsilon_g/m$ and suppose that Assumptions 2.3.1, 2.3.2 and 2.3.3 hold. Let θ_k be the angle between $p_k = -H_k g_k$ and $-g_k$. For a given $q \in (0,1)$, set β_1 as in Lemma 2.3.4, and define the index J of "good iterates" generated by Algorithm 2.1 as*

$$J = \{j \in \mathbb{N} | \cos\theta_j \geq \beta_1\}, \tag{2.38}$$

as well as the set $J_k = J \cap \{0, 1, 2, ..., k-1\}$. Then,

$$|J_k| \geq qk. \tag{2.39}$$

Proof. Since $l > 2\epsilon_g/m$, we know by (2.32) in Lemma 2.3.3 that conditions (2.29) are satisfied for all k. Since

$$\Theta_k = \angle(s_k, B_k s_k) = \angle(p_k, B_k p_k) = \angle(p_k, -g_k) = \theta_k,$$

(2.39) follows from Lemma 2.3.4. □

Having established a lower bound on $\cos\theta_k$ (for the good iterates), the next step is to establish a similar lower bound for $\cos\widetilde{\theta}_k$. To do so, we first prove the following result, which we state in some generality.

Lemma 2.3.5. *Let $p, g_1, g_2 \in \mathbb{R}^d$ be non-zero vectors. Let ϑ_1 be the angle between p and g_1, and ϑ_2 the angle between p and g_2. Assume*

$$\cos\vartheta_1 \geq \beta > 0, \tag{2.40}$$

and that g_1 and g_2 satisfy

$$\|g_1 - g_2\| \leq \epsilon. \tag{2.41}$$

If in addition

$$\frac{\epsilon}{\|g_2\|} \leq \frac{\beta}{4}, \tag{2.42}$$

then

$$\cos \vartheta_2 \geq \frac{\beta}{2}.$$

Proof. From (2.40) we have

$$p^T g_1 \geq \beta \|p\| \|g_1\|,$$

and by (2.41)

$$p^T g_2 \geq \|p\| (\beta \|g_1\| - \epsilon).$$

Hence, by (2.42)

$$\begin{aligned} \cos \vartheta_2 = \frac{p^T g_2}{\|p\| \|g_2\|} &\geq \frac{\beta \|g_1\| - \epsilon}{\|g_2\|} \\ &\geq \frac{\|g_2\| - \epsilon}{\|g_2\|}\beta - \frac{\epsilon}{\|g_2\|} \\ &\geq \left(1 - \frac{\epsilon}{\|g_2\|}\right)\beta - \frac{\beta}{4}. \end{aligned}$$

The bound (2.40) implies that $\beta \leq 1$, and hence

$$\frac{\epsilon}{\|g_2\|} \leq \frac{\beta}{4} \leq \frac{1}{4}.$$

Therefore,

$$\cos\vartheta_2 \geq \left(1 - \frac{\epsilon}{\|g_2\|}\right)\beta - \frac{\beta}{4} \geq \frac{\beta}{2}.$$

□

We also need the following well known result [**48**] about the function decrease provided by the Armijo-Wolfe line search.

Lemma 2.3.6. *Suppose $h : \mathbb{R}^d \to \mathbb{R}$ is a continuous differentiable function with an L-Lipschitz continuous gradient. Suppose $x \in \mathbb{R}^d$, and that $p \in \mathbb{R}^d$ is a descent direction for h at x. Let θ be the angle between $-p$ and $\nabla h(x)$. Suppose $\alpha > 0$ is a step that satisfies the Armijo-Wolfe conditions with parameters $0 < c_1 < c_2 < 1$:*

$$\begin{aligned} h(x+\alpha p) &\leq h(x) + c_1\alpha p^T\nabla h(x) \\ p^T\nabla h(x+\alpha p) &\geq c_2 p^T\nabla h(x). \end{aligned} \tag{2.43}$$

Then

$$h(x+\alpha p) - h(x) \leq -c_1\frac{1-c_2}{L}\cos^2\theta\,\|\nabla h(x)\|^2.$$

Proof. From the second condition in (2.43) we have

$$p^T[\nabla h(x+\alpha p) - \nabla h(x)] \geq (c_2 - 1)p^T\nabla h(x).$$

By Lipschitz continuity,

$$p^T[\nabla h(x+\alpha p) - \nabla h(x)] \leq L\|p\|^2\alpha,$$

and from this it follows that

$$\alpha \geq -\frac{1-c_2}{L}\frac{\nabla h(x)^T p}{\|p\|^2}.$$

Substituting this into the first condition in (2.43) we obtain the desired result. □

We can now show that a fraction of the iterates generated by Algorithm 2.1 produce a decrease in the true objective that is proportional to its gradient.

Theorem 2.3.3. *Suppose that Assumptions 2.3.1, 2.3.2 and 2.3.3 are satisfied, and let* $\{x_k\}$, *and* $\{p_k\}$ *be generated by Algorithm 2.1. Define* β_1 *and* J *as in Corollary 2.3.1. Fix constants* $0 < c_1 < c_2 < 1$, *and choose* $\delta_1, \delta_2, \hat{\delta}_1, \hat{\delta}_2 \in (0,1)$ *such that* $\delta_1 + \delta_2 < c_2 - c_1$ *and* $\hat{\delta}_1 < c_1,\ \hat{\delta}_2 < 1 - c_2$. **If** $k \in J$ *and*

$$\|\nabla\phi(x_k)\| \geq \max\left\{A\frac{\sqrt{M\epsilon_f}}{\beta_1},\ B\frac{\epsilon_g}{\beta_1}\right\}, \tag{2.44}$$

where

$$A = \max\left\{\sqrt{\frac{32}{(1-c_2+\delta_2)\delta_1}},\ \sqrt{\frac{32}{\hat{\delta}_1(1-c_2)}}\right\}$$

$$B = \max\left\{\frac{4(c_1+\delta_1)}{\delta_1},\ \frac{2(1+c_2-\delta_2)}{\delta_2},\ \frac{8}{(1-c_2)},\ \frac{4c_1}{\hat{\delta}_1},\ \frac{2(1+c_2)}{\hat{\delta}_2}\right\},$$

then there exists a stepsize α_k *which satisfies the Armijo-Wolfe conditions for* (f,g) *with parameters* (c_1, c_2), *i.e.,*

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1\alpha_k p_k^T g_k$$

$$p_k^T g(x_k + \alpha_k p_k) \geq c_2 p_k^T g_k,$$

and ***any*** *such stepsize also satisfies the Armijo-Wolfe conditions for* $(\phi, \nabla\phi)$ *with parameters* $(c_1 - \hat{\delta}_1, c_2 + \hat{\delta}_2)$*:*

$$\phi(x_k + \alpha_k p_k) \leq \phi(x_k) + (c_1 - \hat{\delta}_1)\alpha_k p_k^T \nabla\phi(x_k)$$

$$p_k^T \nabla\phi(x_k + \alpha_k p_k) \geq (c_2 + \hat{\delta}_2) p_k^T \nabla\phi(x_k)$$

and in addition,

$$\phi(x_k + \alpha_k p_k) - \phi(x_k) \leq -\frac{(c_1 - \hat{\delta}_1)\left[1 - (c_2 + \hat{\delta}_2)\right]\beta_1^2}{4M} \left\|\nabla\phi(x_k)\right\|^2. \tag{2.45}$$

Proof. Take $k \in J$. By Corollary 2.3.1 we have that $\cos\theta_k \geq \beta_1$. Now, by (2.44)

$$\|\nabla\phi(x_k)\| \geq B\frac{\epsilon_g}{\beta_1} \geq \frac{4(c_1 + \delta_1)}{\delta_1}\frac{\epsilon_g}{\beta_1} \geq 4\frac{\epsilon_g}{\beta_1},$$

which together with Lemma 2.3.5 and Assumption 2.3.2 implies that $\cos\widetilde{\theta}_k \geq \beta_1/2$. Therefore, $p_k = -H_k g_k$ is a descent direction with respect to both g_k and $\nabla\phi(x_k)$, which will enable us to apply Theorems 2.3.1 and 2.3.2.

Before doing so, we need to verify that the assumptions of those two theorems are satisfied, namely (2.21) and (2.25). To see this, note that since we have shown that

$$\cos\theta_k \geq \beta_1, \quad \cos\widetilde{\theta}_k \geq \frac{\beta_1}{2}$$

then from (2.44) it follows that

$$\begin{aligned}
&\|\nabla\phi(x_k)\| \\
&\geq \max\left\{A\frac{\sqrt{M\epsilon_f}}{\beta_1},\ B\frac{\epsilon_g}{\beta_1}\right\} \\
&\geq \max\left\{\sqrt{\frac{32M\epsilon_f}{(1-c_2+\delta_2)\delta_1\beta_1^2}},\ \frac{4(c_1+\delta_1)\epsilon_g}{\delta_1\beta_1},\ \frac{2(1+c_2-\delta_2)\epsilon_g}{\delta_2\beta_1}\right\} \\
&\geq \max\left\{\sqrt{\frac{16M\epsilon_f}{(1-c_2+\delta_2)\delta_1\cos\theta_k\cos\widetilde{\theta}_k}},\ \frac{4(c_1+\delta_1)\epsilon_g}{\delta_1\cos\theta_k},\ \frac{2(1+c_2-\delta_2)\epsilon_g}{\delta_2\cos\theta_k}\right\},
\end{aligned}$$

as well as

$$\begin{aligned}
&\|\nabla\phi(x_k)\| \\
&\geq \max\left\{A\frac{\sqrt{M\epsilon_f}}{\beta_1},\ B\frac{\epsilon_g}{\beta_1}\right\} \\
&\geq \max\left\{\sqrt{\frac{32M\epsilon_f}{\hat{\delta}_1(1-c_2)\beta_1^2}},\ \frac{8\epsilon_g}{(1-c_2)\beta_1},\ \frac{4c_1\epsilon_g}{\hat{\delta}_1\beta_1},\ \frac{2(1+c_2)\epsilon_g}{\hat{\delta}_2\beta_1}\right\} \\
&\geq \max\left\{\sqrt{\frac{16M\epsilon_f}{\hat{\delta}_1(1-c_2)\cos\theta_k\cos\widetilde{\theta}_k}},\ \frac{8\epsilon_g}{(1-c_2)\cos\theta_k},\ \frac{2c_1\epsilon_g}{\hat{\delta}_1\cos\widetilde{\theta}_k},\ \frac{(1+c_2)\epsilon_g}{\hat{\delta}_2\cos\widetilde{\theta}_k}\right\}.
\end{aligned}$$

Therefore, by Theorems 2.3.1 and 2.3.2 there exists a stepsize α_k which satisfies the Armijo-Wolfe conditions for (f, g) with parameters (c_1, c_2), and such α_k also satisfies the Armijo-Wolfe conditions for $(\phi, \nabla\phi)$ with parameters $(c_1 - \hat{\delta}_1, c_2 + \hat{\delta}_2)$. We then apply Lemma 2.3.6 with $h(\cdot) \leftarrow \phi(\cdot)$, $\theta \leftarrow \widetilde{\theta}_k$ and $L \leftarrow M$, Armijo-Wolfe parameters

$(c_1 - \hat{\delta}_1, c_2 + \hat{\delta}_2)$ to obtain

$$\begin{aligned}\phi(x_k + \alpha_k p_k) - \phi(x_k) &\leq -\frac{(c_1 - \hat{\delta}_1)\left[1 - (c_2 + \hat{\delta}_2)\right]}{M} \cos^2 \widetilde{\theta}_k \left\|\nabla\phi(x_k)\right\|^2 \\ &\leq -\frac{(c_1 - \hat{\delta}_1)\left[1 - (c_2 + \hat{\delta}_2)\right]\beta_1^2}{4M} \left\|\nabla\phi(x_k)\right\|^2.\end{aligned}$$

□

The constants A, B, as well as the rate constant in (2.45), do not depend on the objective function or the noise level, but only on the parameters c_1, c_2. There is, nevertheless, some freedom in the specification of A, B and the constant in (2.45) through the choices of $\delta_1, \delta_2, \hat{\delta}_1, \hat{\delta}_2$. From now on, we make a specific choice for the latter four constants, which simplifies Theorem 2.3.3, as shown next.

Corollary 2.3.2. *Suppose Assumptions 2.3.1, 2.3.2 and 2.3.3 are satisfied, and let* $\{x_k\}$, $\{p_k\}$ *be generated by Algorithm 2.1. Define* β_1 *and* J *as in Corollary 2.3.1. Choose* $\delta_1, \delta_2, \hat{\delta}_1, \hat{\delta}_2$ *as*

$$\delta_1 = \frac{c_2 - c_1}{4}, \quad \delta_2 = \frac{c_2 - c_1}{4}, \quad \hat{\delta}_1 = \frac{c_1}{2}, \quad \hat{\delta}_2 = \frac{1 - c_2}{2}. \tag{2.46}$$

If $k \in J$ *and*

$$\|\nabla\phi(x_k)\| \geq \max\left\{A\frac{\sqrt{M\epsilon_f}}{\beta_1},\ B\frac{\epsilon_g}{\beta_1}\right\},$$

where

$$
\begin{aligned}
A &= \max\left\{ \frac{16\sqrt{2}}{\sqrt{(c_2-c_1)(4-c_1-3c_2)}}, \frac{8}{\sqrt{c_1(1-c_2)}} \right\} \\
B &= \max\left\{ \frac{8}{1-c_2}, \frac{8(1+c_1)}{c_2-c_1} + 6 \right\},
\end{aligned} \tag{2.47}
$$

then there exists a stepsize α_k which satisfies the Armijo-Wolfe conditions on (f, g) with parameters (c_1, c_2), i.e.,

$$
\begin{aligned}
f(x_k + \alpha_k p_k) &\leq f(x_k) + c_1 \alpha_k p_k^T g_k \\
p_k^T g(x_k + \alpha_k p_k) &\geq c_2 p_k^T g_k,
\end{aligned}
$$

and any such stepsize also satisfies the Armijo-Wolfe conditions on $(\phi, \nabla\phi)$ with parameters $(c_1/2, c_2/2 + 1)$:

$$
\begin{aligned}
\phi(x_k + \alpha_k p_k) &\leq \phi(x_k) + \frac{c_1}{2} \alpha_k p_k^T \nabla\phi(x_k) \\
p_k^T \nabla\phi(x_k + \alpha_k p_k) &\geq \frac{1+c_2}{2} p_k^T \nabla\phi(x_k),
\end{aligned}
$$

and in addition,

$$
\phi(x_k + \alpha_k p_k) - \phi(x_k) \leq -\frac{c_1(1-c_2)\beta_1^2}{16M} \left\| \nabla\phi(x_k) \right\|^2 .
$$

Proof. We begin by verifying that the choices (2.46) of $\delta_1, \delta_2, \hat{\delta}_1, \hat{\delta}_2$ satisfy the requirements in Theorem 2.3.3. It is clear that $\delta_1, \delta_2, \hat{\delta}_1, \hat{\delta}_2 \in (0, 1)$ since $0 < c_1 < c_2 < 1$. We also have

$$
\delta_1 + \delta_2 = \frac{c_2 - c_1}{2} < c_2 - c_1, \quad \hat{\delta}_1 = \frac{c_1}{2} < c_1, \quad \hat{\delta}_2 = \frac{1-c_2}{2} < 1 - c_2.
$$

Applying Theorem 2.3.3 with the choices (2.46), we have

$$\begin{aligned} A &= \max\left\{\sqrt{\frac{32}{(1-c_2+\delta_2)\delta_1}},\ \sqrt{\frac{32}{\hat{\delta}_1(1-c_2)}}\right\} \\ &= \max\left\{\frac{16\sqrt{2}}{\sqrt{(c_2-c_1)(4-c_1-3c_2)}}, \frac{8}{\sqrt{c_1(1-c_2)}}\right\} \end{aligned}$$

$$\begin{aligned} B &= \max\left\{\frac{4(c_1+\delta_1)}{\delta_1},\ \frac{2(1+c_2-\delta_2)}{\delta_2},\ \frac{8}{(1-c_2)},\ \frac{4c_1}{\hat{\delta}_1},\ \frac{2(1+c_2)}{\hat{\delta}_2}\right\} \\ &= \max\left\{\frac{8}{1-c_2}, \frac{8(1+c_1)}{c_2-c_1}+6\right\}. \end{aligned}$$

Therefore, by Theorem 2.3.3 we know that there exists a stepsize α_k which satisfies the Armijo-Wolfe conditions for (f, g) with parameters (c_1, c_2), and any such stepsize also satisfies the Armijo-Wolfe conditions for $(\phi, \nabla\phi)$ with parameters $(c_1 - \hat{\delta}_1, c_2 + \hat{\delta}_2) = (c_1/2, c_2/2 + 1)$. In addition, we also have

$$\begin{aligned} \phi(x_k + \alpha_k p_k) - \phi(x_k) &\leq -\frac{(c_1 - \hat{\delta}_1)\left[1 - (c_2 + \hat{\delta}_2)\right]\beta_1^2}{4M} \left\|\nabla\phi(x_k)\right\|^2 \\ &= -\frac{c_1(1-c_2)\beta_1^2}{16M} \left\|\nabla\phi(x_k)\right\|^2. \end{aligned}$$

□

2.3.4. Convergence Results

We are ready to state the main convergence results for our algorithm, which is simply Algorithm 2.1 using a lengthening parameter l such that

$$l > 2\epsilon_g/m, \tag{2.48}$$

We begin by establishing some monotonicity results for the true objective function ϕ. Note that since Algorithm 2.1 either computes a zero step (when $\alpha^* = 0$) or generates a new iterate that satisfies the Armijo decrease (2.3), the sequence $\{f(x_k)\}$ is non-increasing.

Theorem 2.3.4. *Suppose Assumption 2.3.2 and 2.3.3 are satisfied, and let* $\{x_k\}$ *be generated by Algorithm 2.1 with* l *satisfying* (2.48). *Define*

$$\xi_k = \min_{i \in [k]} \ \phi(x_i), \qquad \text{where } [k] \stackrel{\text{def}}{=\!=} \ \{i \in \mathbb{N} | 0 \leq i \leq k\}. \tag{2.49}$$

Then $\{\xi_k\}$ *is non-increasing and*

$$\xi_k \leq \phi(x_k) \leq \xi_k + 2\epsilon_f, \quad \forall k \in \mathbb{N}.$$

Proof. By definition, $\{\xi_j\}$ forms a non-increasing sequence, and we noted above that $\{f(x_k)\}$ is also non-increasing and therefore

$$f(x_j) = \min_{i \in [j]} \ f(x_i).$$

By Assumption 2.3.2 we have

$$f(x_i) \leq \phi(x_i) + \epsilon_f.$$

Hence

$$f(x_j) = \min_{i \in [j]} \ f(x_i) \leq \min_{i \in [j]} \ (\phi(x_i) + \epsilon_f) = \min_{i \in [j]} \ \phi(x_i) + \epsilon_f,$$

and recalling again Assumption 2.3.2, we have

$$\phi(x_j) \leq f(x_j) + \epsilon_f \leq \min_{i \in [j]} \ \phi(x_i) + 2\epsilon_f.$$

Since

$$\xi_j = \min_{i \in [j]} \ \phi(x_i) \leq \phi(x_j),$$

we conclude that

$$\xi_j \leq \phi(x_j) \leq \xi_j + 2\epsilon_f.$$

□

The next result shows that, before the iterates $\{x_k\}$ reach a neighborhood of the solution where the error dominates, the sequence $\{\phi(x_k) - \phi^*\}$ converges to the value $2\epsilon_f$ at an R-linear rate. Here ϕ^* denotes the optimal value of ϕ. Note that in this result we assume that the line search is successful at all good iterates, before noise dominates. This is a reasonable assumption since Corollary 2.3.2 guarantees the existence of an interval satisfying the Armijo-Wolfe conditions.

Theorem 2.3.5. [Linear Convergence to $\mathcal{N}_1$] *Suppose Assumptions 2.3.1, 2.3.2 and 2.3.3 are satisfied, and let $\{x_k\}$ be generated by Algorithm 2.1 with l satisfying* (2.48). *Let*

$$\mathcal{N}_1 = \left\{ x \,\middle|\, \|\nabla \phi(x)\| \leq \max \left\{ A\frac{\sqrt{M\epsilon_f}}{\beta_1},\ B\frac{\epsilon_g}{\beta_1} \right\} \right\},$$

where A, B are given in (2.47). *Additionally assume that for all iterates $k \in J$ such that $x_k \notin \mathcal{N}_1$,the algorithm chooses $\alpha_k > 0$ satisfying the Armijo-Wolfe conditions for (f, g). Let*

$$K = \min_k \ \{k \in \mathbb{N} | x_k \in \mathcal{N}_1\}$$

be the index of the first iterate that enters $\mathcal{N}_1$ *(we define* $K = +\infty$ *if no such iterate exists). Then there exists* $\rho \in (0,1)$ *such that*

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

Proof. By definition, we have that $\forall k \leq K-1$

$$\|\nabla\phi(x_k)\| > \max\left\{A\frac{\sqrt{M\epsilon_f}}{\beta_1},\ B\frac{\epsilon_g}{\beta_1}\right\}. \tag{2.50}$$

Choose $0 \leq j \leq k \leq K-1$, and let J be as defined in Corollary 2.3.1. If $j \in J$, then by Corollary 2.3.2 we have

$$\phi(x_{j+1}) - \phi(x_j) \leq -\zeta \|\nabla\phi(x_j)\|^2$$

where

$$\zeta = \frac{c_1(1-c_2)\beta_1^2}{16M}.$$

By Theorem 2.3.4, we have that $\phi(x_j) \leq \xi_j + 2\epsilon_f$, and hence

$$\phi(x_{j+1}) \leq \xi_j + 2\epsilon_f - \zeta \|\nabla\phi(x_j)\|^2.$$

Recalling that

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

and by (2.50) we have

$$\begin{aligned}\zeta \left\|\nabla\phi(x_j)\right\|^2 &\geq \frac{c_1(1-c_2)}{16}A^2\epsilon_f \\ &\geq \frac{c_1(1-c_2)}{16}\left[\frac{8}{\sqrt{c_1(1-c_2)}}\right]^2\epsilon_f \\ &= 4\epsilon_f,\end{aligned}$$

and thus

$$\phi(x_{j+1}) \leq \xi_j - \frac{\zeta}{2}\left\|\nabla\phi(x_j)\right\|^2.$$

Since ϕ is strongly convex by Assumption 2.3.3, we have

$$\left\|\nabla\phi(x_j)\right\|^2 \geq 2m(\phi(x_j)-\phi^*) \geq 2m(\xi_j-\phi^*),$$

thus we have

$$\xi_{j+1} \leq \phi(x_{j+1}) \leq \xi_j - \frac{\zeta}{2}\left\|\nabla\phi(x_j)\right\|^2 \leq \xi_j - m\zeta(\xi_j-\phi^*)$$

i.e.,

$$\xi_{j+1}-\phi^* \leq (1-m\zeta)(\xi_j-\phi^*).$$

The relation above holds if $j \in J$. If $j \notin J$, all we can ascertain is that

$$\xi_{j+1} \leq \xi_j.$$

By Corollary 2.3.1, we have $|[k-1]\cap J| \geq qk$, hence

$$\xi_k - \phi^* \leq (1-m\zeta)^{qk}\ (\xi_0-\phi^*) = \rho^k(\phi(x_0)-\phi^*)$$

where $\rho = (1 - m\zeta)^q$. Since $\phi(x_k) \leq \xi_k + 2\epsilon_f$, we have

$$\phi(x_k) - \phi^* \leq \rho^k(\phi(x_0) - \phi^*) + 2\epsilon_f.$$

□

Remark. It is interesting to consider these results in the noise-free case when $\epsilon_f = \epsilon_g = 0$. Then we may choose the lengthening parameter l to be arbitrarily small so that line 12 in Algorithm 2.1 is never executed, and Algorithm 2.1 reduces to the standard BFGS method with Armijo-Wolfe line search. In this case, we have $\mathcal{N}_1 = \{x^*\}$ in Theorem 2.3.5, which establishes the following linear convergence result:

$$\phi(x_k) - \phi^* \leq \rho^k(\phi(x_0) - \phi^*), \ \forall k \in \mathbb{N}.$$

This result is consistent with established linear convergence for the standard BFGS method (see [**18**, Theorem 3.1]). Without extra assumptions on the linesearch procedure and the continuity of $\nabla^2\phi(x)$, this is the best known rate for BFGS method.

The next result shows that the iterates generated by the algorithm enter the neighborhood $\mathcal{N}_1$ in a finite number of iterations.

Theorem 2.3.6. *Suppose Assumptions 2.3.1, 2.3.2 and 2.3.3 are satisfied. Let* $\{x_k\}$ *be generated by Algorithm 2.1 using* (2.48). *Let* $\mathcal{N}_1$ *and* K *be defined as in Theorem 2.3.5. If in addition we assume that* $\max\{\epsilon_f, \epsilon_g\} > 0$, *then we have*

$$K < +\infty$$

Proof. Suppose, by the way of contradiction, that $K = +\infty$, i.e., that $x_k \notin \mathcal{N}_1$, for all k. Pick arbitrary $\delta > 0$, then by Theorem 2.3.5 we have

$$\phi(x_k) - \phi^* \leq \delta + 2\epsilon_f,$$

for sufficiently large k. On the other hand, by Assumption 2.3.1,

$$\|\nabla\phi(x)\|^2 \leq 2M(\phi(x) - \phi^*),\ \forall x \in \mathbb{R}^d.$$

Hence,

$$\|\nabla\phi(x_k)\|^2 \leq 4M\epsilon_f + 2M\delta.$$

Choose δ sufficiently small such that

$$\|\nabla\phi(x_k)\|^2 \leq 4M\epsilon_f + 2M\delta \leq \left[\max\left\{A\frac{\sqrt{M\epsilon_f}}{\beta_1},\ B\frac{\epsilon_g}{\beta_1}\right\}\right]^2,$$

which is always possible since $A > 2$ and $\beta_1 \in (0, 1)$. Therefore, $x_k \in \mathcal{N}_1$ yielding a contradiction. □

The next result shows that after an iterate has entered the neighborhood $\mathcal{N}_1$, all subsequent iterates cannot stray too far away from the solution in the sense that their function values remain within a band of width $2\epsilon_f$ of the largest function value obtained inside $\mathcal{N}_1$.

Theorem 2.3.7. *Suppose Assumptions 2.3.1, 2.3.2 and 2.3.3 are satisfied. Let* $\{x_k\}$ *be generated by Algorithm 2.1 with the choice* (2.48). *Let* $\mathcal{N}_1$ *and* K *be defined as in*

Theorem 2.3.5, and let

$$\hat{\phi} = \max_{x \in \mathcal{N}_1} \ \phi(x),$$

and

$$\mathcal{N}_2 = \left\{ x | \phi(x) \leq \hat{\phi} + 2\epsilon_f \right\}.$$

Then,

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

Proof. Since ϕ is twice continuously differentiable and strongly convex, $\mathcal{N}_1$ defined in Theorem 2.3.5 is a compact set, so $\hat{\phi}$ is well-defined. By Theorem 2.3.6, $K < \infty$. Choose any $k \geq K$. Since $x_K \in \mathcal{N}_1$ and $k \geq K$, we have

$$\xi_k \leq \xi_K \leq \phi(x_K) \leq \hat{\phi}.$$

Recalling Theorem 2.3.4,

$$\phi(x_k) \leq \xi_k + 2\epsilon_f \leq \hat{\phi} + 2\epsilon_f$$

which shows that $x_k \in \mathcal{N}_2$. □

Finally, we have the following result regarding the lengthening operation. It shows that for all "good iterates" that are sufficiently away from $\mathcal{N}_1$ lengthening is not necessary.

Theorem 2.3.8. *Suppose Assumptions 2.3.1, 2.3.2 and 2.3.3 are satisfied. Let* $\{x_k\}$ *be generated by Algorithm 2.1 with lengthening parameter* l *satisfying* (2.48)*. Let* J *be defined as in Corollary 2.3.1, and* A, B *be defined as* (2.47)*. If* $k \in J$ *and*

$$\|\nabla \phi(x_k)\| \geq \max \left\{ A\frac{\sqrt{M\epsilon_f}}{\beta_1}, \ B\frac{\epsilon_g}{\beta_1}, \ \frac{4lM}{(1-c_2)\beta_1} \right\},$$

then $\|\alpha_k p_k\| \geq l$, meaning that step 12 of Algorithm 2.1 is not executed.

Proof. Since $k \in J$ and

$$\|\nabla\phi(x_k)\| \geq \max\left\{A\frac{\sqrt{M\epsilon_f}}{\beta_1},\ B\frac{\epsilon_g}{\beta_1}\right\},$$

by Theorem 2.3.3 and Corollary 2.3.2 we know that the stepsize α_k satisfies

$$\phi(x_k + \alpha_k p_k) \leq \phi(x_k) + \frac{c_1}{2}\alpha_k p_k^T \nabla\phi(x_k)$$
$$p_k^T \nabla\phi(x_k + \alpha_k p_k) \geq \frac{1+c_2}{2} p_k^T \nabla\phi(x_k).$$

Thus we have a lower bound on α_k:

$$\alpha_k \geq -\frac{1-c_2}{2M}\frac{\nabla\phi(x_k)^T p_k}{\|p_k\|^2}.$$

Then we have

$$\begin{aligned}\|\alpha_k p_k\| &\geq \frac{1-c_2}{2M}\|\nabla\phi(x_k)\|\cos\widetilde{\theta}_k \\ &\geq \frac{(1-c_2)\beta_1}{4M}\|\nabla\phi(x_k)\|.\end{aligned}$$

Since

$$\|\nabla\phi(x_k)\| \geq \frac{4lM}{(1-c_2)\beta_1},$$

we have

$$\|\alpha_k p_k\| \geq \frac{(1-c_2)\beta_1}{4M}\|\nabla\phi(x_k)\| \geq l.$$

□

2.4. Numerical Experiments

We implemented Algorithm 2.1 and tested it on a 4-dimensional quadratic function of the form

$$\phi(x) = \tfrac{1}{2} x^T T x,$$

where the eigenvalues of T are $\lambda(T) = \{10^{-2}, 1, 10^2, 10^4\}$. Thus, the strong convexity parameter is $m = 10^{-2}$ and the Lipschitz constant $M = 10^4$.

The noise in the function $\epsilon(x)$ was computed by uniformly sampling from the interval $[-\epsilon_f, \epsilon_f]$, and the noise in the gradient $e(x)$ by uniformly sampling from the closed ball $\|x\|_2 \leq \epsilon_g$. The maximum noise (or error) level was chosen as $\epsilon_g = \epsilon_f = 1$. We computed the lengthening parameter l in Algorithm 2.1 as $l = 4\epsilon_g/m$, which is twice as large as the lower bound stipulated in Lemma 2.3.3.

The line search implements the standard bisection Armijo-Wolfe search with parameters $c_1 = 0.01, c_2 = 0.5$. If the line search is unable to find an acceptable stepsize within 64 iterations, its is considered to have failed, and we set $\alpha_k = 0$. Algorithm 2.1 terminates if: a) $\|g_k\| \leq 10^{-5}$; or b) 30 consecutive line search failures occur; or c) if Algorithm 2.1 reaches the limit of 60 iterations. The initial iterate is $x_0 = 10^5 \cdot (1, 1, 1, 1)^T$ for which $\|\nabla\phi(x_0)\| \approx 10^9$.

Figures 2.1 and 2.2 plot the results of 20 runs of Algorithm 2.1, all initialized at the vector x_0 given above. In both figures, we indicate the first iteration (in all runs) when the differencing interval was lengthened, i.e., when step 12 of Algorithm 2.1 was executed. We observe from Figure 2.1 that Algorithm 2.1 quickly drives the optimality gap $\phi(x_k) - \phi^*$ to the noise level. Figure 2.3 plots the log of the condition number of

the matrix $H_k^{1/2}\nabla^2\phi(x_k)H_k^{1/2}$ against the iteration number k. For this small dimensional quadratic, the BFGS approximation converges to the true Hessian when errors are not present. Figure 2.3 shows that the Hessian approximation does not deteriorate after the iterates enter the region where noise dominates, illustrating the benefits of the lengthening strategy.

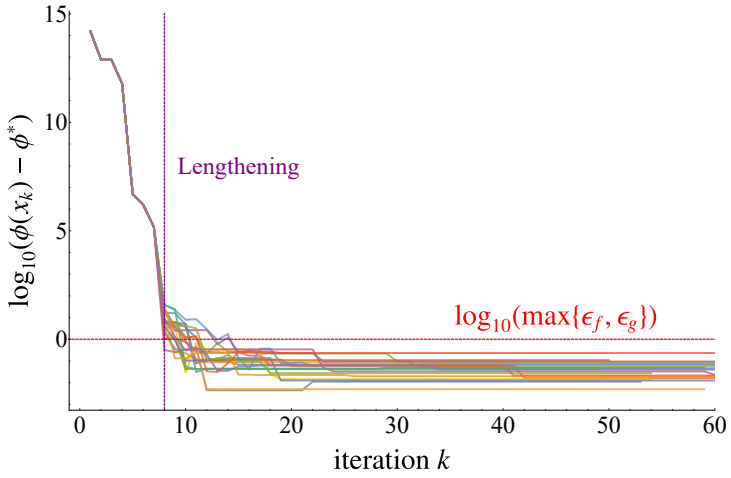

Figure 2.1. Results of 20 runs of Algorithm 2.1. The graph plots the log of the optimality gap for the true function, $\log_{10}(\phi(x_k) - \phi^*)$, against the iteration number k. The horizontal red dashed line corresponds to the noise level $\log_{10} \max\{\epsilon_g, \epsilon_f\} = 0$. The vertical purple dashed line marks the first iteration at which lengthening is performed ($k = 8$).

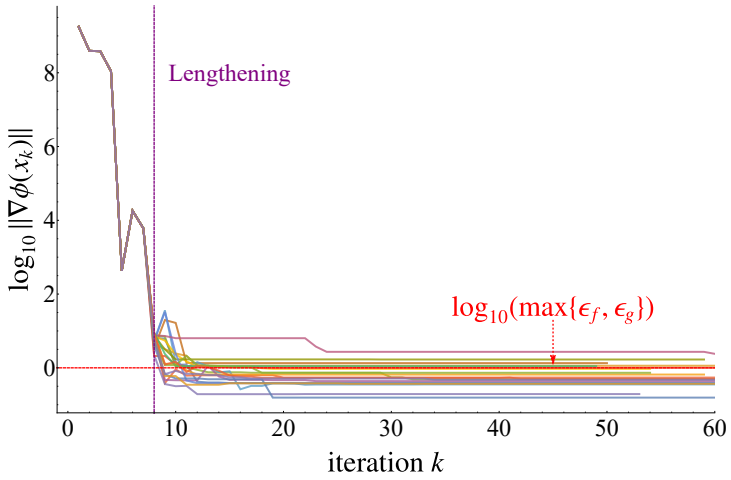

Figure 2.2. Log of the norm of true gradient $\log_{10} \|\nabla\phi(x_k)\|$ against iteration k for 20 runs of Algorithm 2.1. The horizontal red dashed line corresponds to the noise level, and the vertical purple dashed line corresponds to the first iteration at which lengthening is performed.

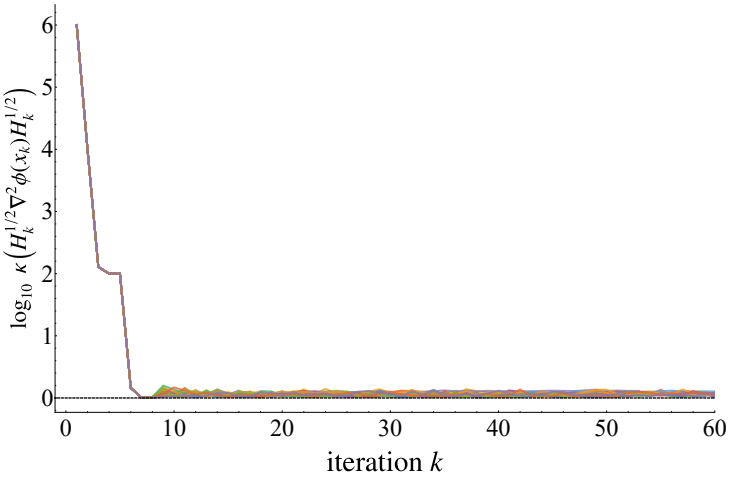

Figure 2.3. Log of the condition number of $H_k^{1/2}\nabla^2\phi(x_k)H_k^{1/2}$ against iteration k. Note that after the iteration reaches the noise level, the Hessian approximation remains accurate.

2.5. Final Remarks

In this chapter, we analyzed the BFGS method when the function and gradient evaluations contain errors. We do not assume that errors diminish as the iterates converge to the solution, or that the user is able to control their magnitude, but that they are always present. Because of this, our analysis focuses on global linear convergence to a neighborhood of the solution, and not on conditions that ensure superlinear convergence — something that would require errors to diminish very rapidly.

In the regime where the gradient $\|\nabla\phi(x)\|$ of the objective function is sufficiently larger than the errors, we might hope for the classical BFGS method to perform well. However, even in that regime, errors can contaminate the Hessian update, and the line search can give conflicting information. In this chapter, we show that a simple modification of the BFGS method inherits the good performance of the classical method (without errors). In particular, we extend one of the hallmark results of BFGS, namely Theorem 2.1 in [**18**], which shows that under mild conditions a large fraction of the BFGS steps are good steps, meaning that they do not tend to be orthogonal to the gradient. We also establish conditions under which an Armijo-Wolfe line search on the noisy function yields sufficient decrease in the true objective function.

The modification of the BFGS method proposed here consists of ensuring that the length of the interval used to compute gradient differences is large enough so that differencing is stable. Specifically, if the line search indicates that the size of the latest step is not large enough compared to the size the gradient error, then the corrections pairs (s_k, y_k) used to update the BFGS matrix are modified. Instead of using s_k as the differencing interval, we lengthen it and compute gradient differences based on the end points of the

elongated interval. This allows us to establish convergence results to a neighborhood of the solution where progress is not possible, along the lines of Nedic and Bertsekas [**46**]. An additional feature of our modified BFGS method is that, when the iterates enter the region where errors dominate, the Hessian approximation does not get corrupted.

The numerical results presented here are designed to verify only the behavior predicted by the theory. In our implementation of Algorithm 2.1, we assume that the size of the errors and the strong convexity parameter m are known, as this helps us determine the size of the lengthening parameter l, although as discussed immediately following Lemma 2.3.3, the algorithm can be modified to avoid the need for knowledge of m without disturbing the analysis.

In the next chapter, we consider a practical implementation of our algorithm that estimates l adaptively and is able to deal with nonconvexity, and that describes a limited memory version of the algorithm. The theory presented in this chapter provides the foundations for the design of such a practical algorithm.

CHAPTER 3

A Noise-Tolerant Quasi-Newton Algorithm for Unconstrained Optimization

3.1. Introduction

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

This chapter builds upon the theoretical results of Xie et al. [**58**] who show that by incorporating a lengthening procedure, the BFGS method enjoys global convergence guarantees to a neighborhood of the solution for strongly convex functions. However, the algorithm proposed in [**58**] is not practical as it requires knowledge of the strong convexity parameter m of the objective function, which is normally not known. An overestimate of m may lead to an unstable iteration, whereas an underestimate can slow down convergence. The quasi-Newton algorithms proposed in this chapter compute the lengthening parameter adaptively without the need for exogenous function information;

they are designed for solving general nonlinear optimization problems and are supported by a convergence analysis for strongly convex objectives. A distinctive feature of our approach is the use of a new line search procedure that works in conjunction with the lengthening technique introduced in this chapter.

The problem under consideration is

$$\min_{x \in \mathbb{R}^d} \ \phi(x) \tag{3.1}$$

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

$$f(x) = \phi(x) + \epsilon(x), \qquad g(x) = \nabla\phi(x) + e(x), \tag{3.2}$$

where the scalar ϵ and the vector e model the errors. We will consider the setting where the errors are bounded and the bounds are either known or estimated through an auxiliary procedure, such as `ECNoise` or pointwise sample variance estimation [**43**]. Specifically, we assume $|\epsilon(x)| \leq \epsilon_f$ and $\|e(x)\|_2 \leq \epsilon_g$ for all $x \in \mathbb{R}^d$, and that the algorithm has access to ϵ_f and ϵ_g.

Problems of this type arise in many practical applications, including when the noise is computational or adversarial. For example, in PDE-constrained optimization, the objective function often contains computational noise created by an inexact linear system solver [**43**], adaptive grids [**1**], or other internal computations. In those applications, the optimization method may not be able to control the size of the errors. In other cases, errors are due to stochastic noise, which can be caused, for example, by an intermediate Monte Carlo simulation [**19**]. In these cases, errors may be controllable via Monte

Carlo sampling. Error in the gradient can also be inherited from noise in the function within derivative-free optimization while employing gradient approximations based on finite-differencing, interpolation, or smoothing [**6, 7, 28, 29, 44, 47**]. In this case, the gradient errors can be controlled by the choice of the finite-difference interval, but can only be diminished to a certain extent under the presence of function noise. We note, however, that there are applications where noise is not bounded or where the bounds ϵ_f, ϵ_g depend on x, in which case the methods proposed here cannot be directly applied.

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

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

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

To provide more concrete numerical evidence for the need to bolster the BFGS method, we illustrate in Figure 3.1 the solution of the `ARWHEAD` problem [**30**] in which independent random noise uniformly distributed on $[-10^{-3}, 10^{-3}]$ is introduced to each component of the gradient. One can observe a very large increase in the condition number of the BFGS matrix that is unseen when noise is removed. This shows that the Hessian approximation

is corrupted, and an examination of the run indicates that the line search gives rise to tiny steps once this has occurred. The `ARWHEAD` problem is chosen because it is easily solved yet clearly illustrates the instability of the BFGS matrix under the presence of noisy updates; we revisit this example in §3.5.1.

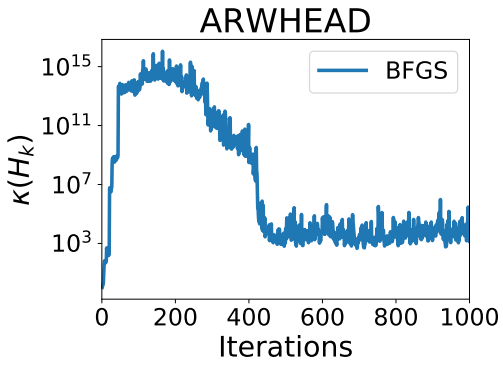

Figure 3.1. The condition number of the BFGS matrix $\kappa(H_k)$ against the number of iterations on the ARWHEAD problem with added noise.

The literature of the BFGS method with inaccurate gradients includes the implicit filtering method of Kelley et al. [**22, 36**], which assumes that noise can be diminished at will at any iteration. Dennis and Walker [**26**] and Ypma [**59**] study bounded deterioration properties and local convergence of quasi-Newton methods with errors, when started near the solution with a Hessian approximation that is close to the exact Hessian. Barton [**2**] and Berahas, et al. [**5**] propose implementations of the BFGS method and L-BFGS method in which gradients are computed by an appropriate finite differencing technique, assuming that the noise level in the function evaluation is known. There has recently been some interest in designing quasi-Newton methods for machine learning applications using stochastic approximations to the gradient [**9, 15, 17, 32, 45, 56**]. These papers avoid potential difficulties with BFGS or L-BFGS updating by assuming that the quality

of gradient differences is sufficiently controlled, and as a result, the analysis follows similar lines as for their classical counterparts. The work that is most relevant to this chapter is by Xie et al. [**58**], who introduce the lengthening technique and establish conditions under which a steplength satisfying the Armijo-Wolfe line search conditions exists.

The contributions of this work are as follows: i) we propose practical extensions of the BFGS and L-BFGS methods for nonlinear optimization that are capable of dealing with noise by employing a new line search/lengthening technique that stabilizes the quasi-Newton update. This strategy relies on the noise control condition (3.11) introduced in this chapter; ii) we provide a convergence analysis for the proposed method for strongly convex objective functions based on the properties the noise control condition instead of assuming knowledge of the strong convexity parameter, as is done in [**58**]; iii) we describe implementations of the methods in full detail, and present extensive numerical results that suggest that our approach is robust for certain classes of noisy optimization problems.

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

3.2. The Algorithm

The BFGS and L-BFGS methods for minimizing ϕ, when only noisy observations (3.2) of the function and gradient are available, have the form

$$x_{k+1} = x_k - \alpha_k H_k g(x_k), \tag{3.3}$$

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

$$(s_k, y_k) = (x_{k+1} - x_k, g(x_{k+1}) - g(x_k)) \tag{3.4}$$

$$= (\alpha_k p_k, g(x_k + \alpha_k p_k) - g(x_k)), \tag{3.5}$$

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

$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T, \qquad \text{where } \rho_k = 1/y_k^T s_k. \tag{3.6}$$

The L-BFGS method stores the past t curvature pairs and computes the matrix-vector product $H_k g_k$ via a two-loop recursion, with memory and computational complexity that is linear with respect to the problem dimension d [**40**]. For both methods, a line search ensures that $y_k^T s_k > 0$, guaranteeing that the update (3.6) is well defined.

As discussed in the previous section, the difference in gradients $g(x_k+\alpha_k p_k)-g(x_k)$ may be dominated by noise, rendering the curvature information inaccurate and potentially malign. To safeguard against this, Xie et al. [**58**] introduced a lengthening operation that ensures that meaningful curvature information is being collected. Specifically, they redefine the curvature pair by

$$(s_k, y_k) = (\beta_k p_k, g(x_k + \beta_k p_k) - g(x_k)), \tag{3.7}$$

where $\beta_k \geq \alpha_k$ is a sufficiently large lengthening parameter. The theoretical analysis in [**58**] states that setting $\beta_k = O(\epsilon_g/m\|p_k\|)$ ensures linear convergence to a neighborhood of the solution for strongly convex problems, where m is the strong convexity parameter

and ϵ_g is an upper bound on the norm of the gradient noise, i.e.,

$$\|g(x) - \nabla\phi(x)\|_2 = \|e(x)\|_2 \leq \epsilon_g \qquad \forall x \in \mathbb{R}^d. \tag{3.8}$$

However, the analysis in [**58**] does not directly yield an implementable algorithm, as the parameter m is generally not known in practice. Furthermore, [**58**] does not propose a practical line search procedure for finding a steplength that satisfies the Armijo-Wolfe conditions—although it does establish the existence of such a steplength.

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

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k g(x_k)^T p_k \qquad \text{(Armijo condition)} \tag{3.9}$$

$$g(x_k + \alpha_k p_k)^T p_k \geq c_2 g(x_k)^T p_k \qquad \text{(Wolfe condition)} \tag{3.10}$$

$$(g(x_k + \beta_k p_k) - g(x_k))^T p_k \geq 2(1 + c_3)\epsilon_g \|p_k\| \qquad \text{(noise control)} \tag{3.11}$$

where $0 < c_1 < c_2 < 1$ and $c_3 > 0$. Here and throughout the chapter, $\|\cdot\|$ denotes the Euclidean norm. The Armijo-Wolfe conditions (3.9)–(3.10) ensure that the steplength α_k that is taken by the algorithm is not too short and yields sufficient decrease on the (noisy) objective function, while the noise control condition (3.11) on β_k is designed so that the difference in the observed directional derivatives is sufficiently large so as not to be dominated by noise. A motivation for (3.11) and a discussion of its salient properties are given below.

To satisfy the three conditions above one could find a steplength α_k that satisfies (3.9)-(3.10), and if (3.11) holds for $\beta_k = \alpha_k$, then set $\beta_k \leftarrow \alpha_k$. Otherwise, one can search for $\beta_k > \alpha_k$ to satisfy (3.11). In practice, we employ a different strategy described in section 3.4.1 to achieve similar objectives.

The outline of the proposed method is given in Algorithm 3.1.

Algorithm 3.1: *Outline of Noise-Tolerant BFGS and L-BFGS Methods*

1: **Input:** function $f(\cdot)$ and gradient $g(\cdot)$; noise level in gradient ϵ_g; initial iterate x_0 and Hessian approximation $H_0 \succ 0$;
2: **for** $k = 0, 1, 2, ...$ **do**
3: Compute $p_k = -H_k g(x_k)$ by matrix-vector multiplication (BFGS) or two-loop recursion [**48**] (L-BFGS);
4: Perform a line search to obtain α_k satisfying (3.9) and (3.10); if the line search fails, then compute α_k such that $f(x_k + \alpha_k p_k) \leq f(x_k)$;
5: Take the step $x_{k+1} = x_k + \alpha_k p_k$;
6: Perform a lengthening procedure to obtain β_k satisfying (3.11);
7: Compute the curvature pair (s_k, y_k) using β_k, as in (3.7);
8: Update the Hessian approximation H_k by (3.6) (BFGS) or update set $\{(s_i, y_i)\}$ of curvature pairs (L-BFGS);
9: **end for**

The Armijo-Wolfe line search is guaranteed to find a steplength α_k that satisfies conditions (3.9)-(3.10) only when the gradient is sufficiently large relative to the noise level; otherwise p_k is not guaranteed to be a descent direction. To handle this case, a line search failure occurs when a maximum number of trial points is computed without satisfying (3.9) and (3.10). The algorithm requires an estimate of the noise level ϵ_g, which can be obtained through sampling or through the Hamming procedure described in [**44**]. The

main remaining ingredient in this algorithm is a description of a procedure for computing α_k and β_k in step 3 and 5. This will be discussed in §3.4.2.

3.2.1. Motivation of the Noise Control Condition (2.9)

We first note that the Wolfe condition (3.10) alone does not ensure that the BFGS update is productive in the noisy setting. Even though (3.10) guarantees that

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

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

Let us, in contrast, observe the effect of the noise control condition (3.11). We have

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

by (3.8). Combining this with (3.11) we have

$$(\nabla\phi(x_k+\beta_k p_k) - \nabla\phi(x_k))^T p_k \geq 2c_3\epsilon_g\|p_k\|, \tag{3.12}$$

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

$$\tilde{y}_k = \nabla\phi(x_k + \beta_k p_k) - \nabla\phi(x_k), \tag{3.13}$$

and recall from (3.7) that $s_k = \beta_k p_k$, we can write (3.12) as

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

We can also establish a relationship between the observed and true curvature along the step s_k. In particular, if $\beta_k > 0$ satisfies the noise control condition (3.11) and (3.8) holds, then

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

which implies that

$$\left(1 + \frac{1}{1+c_3}\right)^{-1} s_k^T \tilde{y}_k \leq s_k^T y_k \leq \left(1 - \frac{1}{1+c_3}\right)^{-1} s_k^T \tilde{y}_k. \tag{3.14}$$

This result shows that when condition (3.11) is satisfied, the noisy curvature pair (s_k, y_k) is a good approximation to the true curvature pair $(s_k, \tilde{y}_k)$, with the parameter c_3 trading off the quality of this approximation with the locality of the curvature information being collected (in the sense that β_k may be excessively large if c_3 is chosen to be large).

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

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

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

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

$$\beta = O(\epsilon_g / m\|p_k\|) \tag{3.15}$$

stipulated by Xie et al. [**58**] in their convergence analysis of the BFGS method with errors for strongly convex functions. If ϕ is m-strongly convex, then

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

Therefore, by our assumption, we have

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

Therefore it suffices to ensure that

$$m\beta_k\|p_k\| - 2\epsilon_g \geq 2(1 + c_3)\epsilon_g, \text{ i.e. } \beta_k \geq \frac{2(2 + c_3)\epsilon_g}{m\|p_k\|}, \tag{3.16}$$

to satisfy (3.11).

Remark 1. It is natural to ask whether there are less expensive alternatives to the lengthening strategy mentioned above. The noise control condition (3.11) offers the possibility of skipping the BFGS update when it is not satisfied. We describe this approach and test it in §3.5. Another possibility is to use Powell damping [**48**, chapter 18], but we consider this to be somewhat dangerous, as it would involve repeatedly introducing spurious information in the Hessian approximation without much safeguard.

3.3. Convergence Analysis

Xie et al. [**58**] established convergence results for the BFGS method using a lengthening strategy designed to cope with errors in the function and gradient. They assume the lengthening parameter satisfies $\beta_k\|p_k\| \geq 2\epsilon_g/m$. This leaves open the question of how to estimate the strong convexity parameter m in practice so that the convergence results in [**58**] still hold.

In this chapter, we bypass this thorny issue and propose the lengthening strategy based on the noise control condition (3.11), which employs an estimate of the noise level of the gradient ϵ_g, but does not require knowledge of m. We now establish conditions under which Algorithm 3.1 is linearly convergent to a neighborhood of the solution determined by the noise level. We make the following assumption about the underlying function ϕ, which is standard in the analysis of quasi-Newton methods.

Assumption 3.3.1. *The function ϕ is m-strongly convex and has M-Lipschitz continuous gradients, i.e., there exist constants $0 < m \leq M$ such that*

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

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

Assumption 3.3.2. *There are constants $\epsilon_g \geq 0$ and $\epsilon_f \geq 0$ such that*

$$\|\nabla\phi(x) - g(x)\| \leq \epsilon_g, \ \forall x \in \mathbb{R}^d, \quad \textit{and} \tag{3.17}$$

$$|\phi(x) - f(x)| \leq \epsilon_f, \ \forall x \in \mathbb{R}^d. \tag{3.18}$$

Byrd and Nocedal [**18**] showed that if all curvature pairs (s_k, y_k) satisfy

$$\frac{s_k^T y_k}{s_k^T s_k} \geq \widehat{m}, \quad \frac{y_k^T y_k}{s_k^T y_k} \leq \widehat{M}, \quad \forall k \in \mathbb{N}, \tag{3.19}$$

for some constants $0 < \widehat{m} \leq \widehat{M}$, then most of the iterates generated by the (classical) BFGS method are "good iterates" in the sense that the angle between the search direction and the steepest direction is bounded away from orthogonality. This fact is used in [**18**] to establish convergence of the BFGS algorithm with various types of line searches for strongly convex functions.

The first step in our analysis consists of showing that bounds of the form (3.19) are satisfied for both the BFGS and L-BFGS versions of our noise tolerant Algorithm 3.1, due to the role of the noise control condition (3.11). For convenience, we summarize the notation introduced in the previous section:

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

and therefore the noise control condition can be written as

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

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

Notation. So far we let H_k denote the BFGS approximation of the inverse Hessian. The classical analysis of the BFGS method analyzes, however, the direct Hessian approximation B_k defined as $B_k^{-1} = H_k$ [**48**]. Therefore, some of the results quoted from [**58**], are stated in terms of B_k.

Lemma 3.3.1. *Suppose that Assumptions 3.3.1 and 3.3.2 hold and that $s_k \neq 0$ is chosen such that*

$$s_k^T \left[g(x_k + s_k) - g(x_k)\right] \geq c\,\epsilon_g \|s_k\|, \tag{3.20}$$

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

$$\frac{s_k^T y_k}{s_k^T s_k} \geq \frac{c}{c+2} m, \qquad \frac{y_k^T y_k}{s_k^T y_k} \leq \frac{c}{c-2} M. \tag{3.21}$$

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

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

In addition, since $\|\tilde{y}_k - y_k\| \leq 2\epsilon_g$ and by Assumption 3.3.1 we have

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

Combining these two inequalities, we obtain

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

which proves the first inequality in (3.21).

For the second bound in (3.21), first note that $\|y_k\| \leq M\|s_k\| + \|\tilde{y}_k - y_k\| \leq M\|s_k\| + 2\epsilon_g$. Therefore,

$$\|s_k\| \left(M\|s_k\| + 2\epsilon_g\right) \geq \|s_k\|\|y_k\| \geq s_k^T y_k \geq c\epsilon_g \|s_k\|, \tag{3.22}$$

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

$$\|s_k\| \geq (c-2)\frac{\epsilon_g}{M}. \tag{3.23}$$

Since ϕ is m-strongly convex with M-Lipschitz continuous gradients, by [**10**, Proposition 6.1.9 (b)] we have

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

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

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

By re-arranging the terms, we get

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

which is equivalent to

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

Consequently

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

i.e.,

$$\|y_k\|^2 - (M+m)s_k^T y_k + \left(\frac{M+m}{2}\right)^2 \|s_k\|^2$$
$$\leq \left(\frac{M-m}{2}\right)^2 \|s_k\|^2 + 2(M-m)\|s_k\|\epsilon_g + 4\epsilon_g^2.$$

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

$$\frac{y_k^T y_k}{s_k^T y_k} \leq (M+m) + \frac{(2\epsilon_g + M\|s_k\|)(2\epsilon_g - m\|s_k\|)}{s_k^T y_k}. \tag{3.24}$$

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

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

From (3.22) we know that

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

therefore,

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

Combining this with the lower bound $\|s_k\| \geq (c-2)\epsilon_g/M$ given in (3.23), we have

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

Case 2: if $2\epsilon_g - m\|s_k\| \geq 0$, then we have from (3.24) and (3.20)

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

The right hand side increases as $\|s_k\|/\epsilon_g$ decreases, hence setting $\|s_k\|$ to the lower bound given in (3.23), we have

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

This proves the second inequality. □

As mentioned above, if we set $c = 2(1+c_3)$ in (3.20), we obtain the noise control condition (3.11). Therefore, we have the following guarantee on the curvature pairs generated by Algorithm 3.1:

$$\frac{s_k^T y_k}{s_k^T s_k} \geq \widehat{m} = \frac{1+c_3}{2+c_3}m, \quad \frac{y_k^T y_k}{s_k^T y_k} \leq \widehat{M} = \left(1 + \frac{1}{c_3}\right)M, \ k = 0, 1, 2, ... \tag{3.25}$$

To continue using the results in [**18**] we define, for any $\gamma > 0$, the index of "good iterates" $J(\gamma)$ as

$$J(\gamma) = \{k \in \mathbb{N} | \cos\theta_k \geq \gamma\}, \tag{3.26}$$

where $\cos\theta_k$ is the angle between $p_k = -H_k g_k$ and $-g_k$. The following lemma uses the bounds (3.25) to show that that for some values γ, the set $J(\gamma)$ contains a fraction of the iterates.

Lemma 3.3.2. *Let $\{x_k\}$, $\{p_k\}$ be generated by Algorithm 3.1, using either the full-BFGS or L-BFGS variant. Then for any $0 < q < 1$, there exists $\gamma > 0$ such that*

$$|J(\gamma) \cap [0, k-1]| \geq qk, \tag{3.27}$$

where $J(\gamma)$ is defined by (3.26).

Proof. For the full-BFGS variant of Algorithm 3.1, since we have shown that (3.25) holds, Theorem 2.1 in [**18**] guarantees that for any $0 < q < 1$, there exists $\gamma_F > 0$ such that

$$|J(\gamma_F) \cap [0, k-1]| \geq qk. \tag{3.28}$$

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

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

Now we apply techniques developed in [**18**]. For any positive definite matrix B, let

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

Since all curvature pairs $\{(s_k, y_k)\}$ satisfy (3.25), by [**18**, (2.9)] we have

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

Therefore, we have

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

By [**18**, (2.7)], we have

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

By (3.25) and the Cauchy-Schwarz inequality,

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

hence,

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

which implies that

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

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

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

Therefore, we have

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

i.e.,

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

which finishes the proof. □

By the discussions above, for both full-BFGS and L-BFGS variants of Algorithm 3.1, we can choose a fixed $q^* \in (0,1)$ and find $\gamma^* > 0$ such that

$$|J(\gamma^*) \cap [0, k-1]| \geq q^* k, \ \forall k \in \mathbb{N}; \tag{3.29}$$

i.e., such that a fraction of iterates are guaranteed to be good iterates. From now on, let us fix the choice q^* and γ^*. Using the above results together with the analysis in [**58**] we arrive at the following convergence result.

Theorem 3.3.1. *Suppose that Assumptions 3.3.1 and 3.3.2 hold. Let* $\{x_k\}$ *be generated by Algorithm 3.1, using either L-BFGS or standard BFGS. Fix* $q^* \in (0,1)$ *and choose* $\gamma^* > 0$ *such that* (3.29) *holds. Define*

$$\mathcal{N}_1 = \left\{ x \;\middle|\; \|\nabla \phi(x)\| \leq \max\left\{ A\frac{\sqrt{M\epsilon_f}}{\gamma^*}, B\frac{\epsilon_g}{\gamma^*} \right\} \right\}, \tag{3.30}$$

and

$$\mathcal{N}_2 = \left\{ x \;\middle|\; \phi(x) \leq 2\epsilon_f + \max_{y \in \mathcal{N}_1} \ \phi(y) \right\} \supseteq \mathcal{N}_1, \tag{3.31}$$

where

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

Let

$$K = \min_k \{k \in \mathbb{N} \mid x_k \in \mathcal{N}_1\} \tag{3.32}$$

be the index of the first iterate that enters $\mathcal{N}_1$*. Assume that for all iterates* $k \in J(\gamma^*)$ *such that* $x_k \notin \mathcal{N}_1$ *the line search procedure finds* α_k *satisfying* (3.9)–(3.10)*. Then there exists* $\rho \in (0, 1)$ *such that*

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

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

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

Proof. Note that Algorithm 3.1 differs from Algorithm 2.1 of [**58**], only in the quasi-Newton updating strategy and lengthening procedure. This implies that the results through Theorem 3.5 of [**58**] concerning the existence of an Armijo-Wolfe stepsize, also apply to Algorithm 2.1 of this chapter, since the proofs of these these results do not depend on the update used. In Lemma 3.3.1 of this chapter we showed that the lengthening procedure in step 5 of Algorithm 3.1 guarantees bounds on $(s_k^T y_k / s_k^T s_k)$ and $(y_k^T y_k / s_k^T y_k)$

such as those of Lemma 3.8 of [**58**]. Using these bounds we established Lemma 3.3.2 whose results are identical to those of Corollary 3.10 in [**58**], with γ^* replacing β_1. With that change, the rest of the results of [**58**], including Theorems 3.16–3.18, hold for Algorithm 3.1 of this chapter, proving the theorem. □

Theorem 3.3.1 states that the iterates generated by Algorithm 3.1 converge linearly to a neighborhood of the solution $\mathcal{N}_1$, whose size depends on the noise levels ϵ_f, ϵ_g; the iterates will enter $\mathcal{N}_1$ in finite number of iterations, and will remain in a larger neighborhood $\mathcal{N}_2$ thereafter.

3.4. A Practical Algorithm

In order to implement Algorithm 3.1, we need to design a practical procedure for computing the steplength α_k and the lengthening parameter β_k. This can be done in various ways, and in this section we present a technique that has performed well in practice. After describing this algorithm in detail, we present several heuristics designed to improve its practical performance.

3.4.1. Two-Phase Line Search and Lengthening Procedure

Algorithm 3.1 and the convergence analysis of the previous section require that α_k and β_k satisfy conditions (3.9), (3.10) and (3.11). We now propose a procedure for computing these quantities.

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

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k g(x_k)^T p_k \tag{3.33}$$

$$g(x_k + \alpha_k p_k)^T p_k \geq c_2 g(x_k)^T p_k \tag{3.34}$$

$$|(g(x_k + \alpha_k p_k) - g(x_k))^T p_k| \geq 2(1 + c_3)\epsilon_g \|p_k\|, \tag{3.35}$$

where $0 < c_1 < c_2 < 1$ and $c_3 > 0$. Observe that (3.35) and the Wolfe condition (3.34) imply the noise control condition (3.11) employed so far in the chapter. We incorporate the absolute value in (3.35) in order to introduce a symmetric noise condition that can be used to determine when to adapt α_k and β_k independently. If $\epsilon_f = \epsilon_g = 0$, then we can guarantee that the initial phase will reduce to the standard Armijo-Wolfe line search, as we describe below.

The initial phase is done using the logic of the standard bisection search: backtracking if the Armijo condition is not satisfied, and advancing if the Armijo condition is satisfied and the Wolfe condition is not, but with one important modification. If the Armijo condition (3.33) is satisfied, we will check (3.35) prior to checking the Wolfe condition (3.34).

If at *any* iteration of the line search the noise control condition (3.35) is not satisfied or if the line search has performed more than the allowed number (N_{split}) of iterations, then the initial phase is terminated and the second phase, which we call the *split phase*, is triggered. In this phase, α_k and β_k are updated independently from each other. The steplength α_k is updated via the standard Armijo backtracking line search while the

lengthening parameter β_k is lengthened independently until the conditions

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k g(x_k)^T p_k \tag{3.36}$$

$$(g(x_k + \beta_k p_k) - g(x_k))^T p_k \geq 2(1 + c_3)\epsilon_g \|p_k\| \tag{3.37}$$

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

The two-phase line search (without heuristics) is presented in Algorithms 3.2 and 3.3. For completeness, we also present the pseudocode for the complete practical algorithm in 3.4.

Algorithm 3.2: *Two-Phase Armijo-Wolfe Line Search and Lengthening: Initial Phase*

Input: functions $f(\cdot)$ and $g(\cdot)$; noise level ϵ_g; current iterate x; search direction p; initial steplength $\alpha = 1$; constants $0 < c_1 < c_2 < 1$, $c_3 > 0$; maximum number of line search iterations before split N_{split}

$l \leftarrow 0$, $u \leftarrow \infty$; ▷ Initialize brackets for bisection

for $i = 0, 1, 2, ..., N_{\text{split}} - 1$ **do**

  if $f(x + \alpha p) > f(x) + c_1 \alpha g(x)^T p$ **then** ▷ Armijo condition fails

    $u \leftarrow \alpha$;

    $\alpha \leftarrow (u + l)/2$; ▷ Backtrack

  else if $|(g(x + \alpha p) - g(x))^T p| < 2(1 + c_3)\epsilon_g \|p\|$ **then** ▷ Noise control condition fails

    Break (for loop)

  else if $g(x + \alpha p)^T p < c_2 g(x)^T p$ **then** ▷ Wolfe condition fails

    $l \leftarrow \alpha$;

    if $u = \infty$ **then** ▷ Advance

      $\alpha \leftarrow 2\alpha$;

    else

      $\alpha \leftarrow (u + l)/2$;

    end if

  else ▷ Satisfies all conditions

    $\beta \leftarrow \alpha$;

    Return α, β;

  end if

end for

$\alpha, \beta \leftarrow \text{SplitPhase}(f, g, \epsilon_g, x, p, \alpha, \beta)$; ▷ Enter split phase

Return α, β;

Algorithm 3.3: *Split Phase*

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

Algorithm 3.4: *Complete Practical Noise-Tolerant BFGS and L-BFGS Methods*

Input: function $f(\cdot)$ and gradient $g(\cdot)$; noise level in function ϵ_f, noise level in gradient ϵ_g; initial iterate x_0 and Hessian approximation $H_0 \succ 0$;
for $k = 0, 1, 2, ...$ **do**
    Compute $p_k = -H_k g(x_k)$ by matrix-vector multiplication (BFGS) or two-loop recursion [**48**] (L-BFGS);
    Perform two-phase Armijo-Wolfe line search (Algorithms 3.2 and 3.3) to find α_k and β_k;
    if α_k satisfies (3.33) **then**
        Compute $x_{k+1} = x_k + \alpha_k p_k$;
    end if
    if β_k satisfies (3.11) **then**
        Compute curvature pair $(s_k, y_k) = (\beta_k p_k, g(x_k + \beta_k p_k) - g(x_k))$;
        Update H_k by (3.6) (BFGS) or update set $\{(s_i, y_i)\}$ of curvature pairs (L-BFGS);
    end if
end for

By the design of the two-phase line search, our algorithm behaves the same as the standard (L-)BFGS algorithm (without interpolation) for non-noisy problems as long as N_{split} is sufficiently large because the split phase will never occur. In particular, if $\epsilon_g = 0$, then condition 3.35 will always be satisfied by any α_k and therefore the initial phase reduces to the standard Armijo-Wolfe line search. However, unlike the deterministic setting, the two-phase line search may not be guaranteed to find α_k and β_k under certain scenarios. When the iteration has reached the region where errors are large relative to the gradient, the backtracking line search may fail to find α_k; this is to be expected. A more subtle case is when the function is exceedingly flat along the search direction p_k so that even for a large β the function exhibits insufficient change in curvature; in this case the lengthening procedure may fail to find an appropriate β_k. To safeguard against both of these cases, the algorithm will terminate if it reaches a maximum number of line search iterations.

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

A second alternative, given in Algorithm 3.1, is the approach employed by Xie et al. [**58**], who first solve for a steplength α_k that satisfies the Armijo-Wolfe conditions (3.33)-(3.34), then lengthen $\beta_k \geq \alpha_k$ until β_k satisfies the noise control condition (3.35). However,

we have found experimentally that performing an Armijo-Wolfe line search attempting to find a steplength that satisfies the Armijo-Wolfe conditions in the presence of noise can be costly in terms of function and gradient evaluations because the Armijo-Wolfe line search may be fooled for many iterations in the presence of moderate to large noise relative to the gradient. In particular, enforcing the Wolfe condition on the steplength when the gradient is dominated by noise may lead to ill-advised or unnecessary changes to the steplength. Rather than doing this, we opt to split the computations of β and α earlier, as done in Algorithm 3.2 using (3.35) as a means to detect when to split and consider the Wolfe condition unreliable.

3.4.2. Heuristics

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

I. Relaxation of Armijo Condition. The last term in the Armijo condition (3.33) ensures sufficient descent, but is useful only if the quantities involved are reliable; otherwise it is best to dispense with this term. To see this, consider the term $g(x_k)^T p_k$. Although $g(x_k)^T p_k = -g(x_k)^T H_k g(x_k) < 0$ since H_k is positive definite, this quantity could still be dominated by noise. If $g(x_k)^T p_k < -\epsilon_g \|p_k\|$, we can guarantee that $\nabla\phi(x_k)^T p_k < 0$, ensuring that p_k is a descent direction with respect to the true objective function. If instead we have that $g(x_k)^T p_k \geq -\epsilon_g \|p_k\|$, it is not guaranteed that we can make progress on the true objective function along p_k. In this case, we will consider the gradient estimate unreliable and dispense the sufficient decrease term, instead relaxing the condition to only enforce simple decrease $f(x_k + \alpha p_k) < f(x_k)$.

Another feature that is useful when the algorithm reaches a region where the noise in the function is large relative to the objective function is to relax the Armijo condition (3.33) by adding $2\epsilon_f$ to the right hand side. This relaxation will be done only after the first attempt at satisfying the standard Armijo condition fails. If p_k is a descent direction with respect to ϕ, which is ensured when $g(x_k)^T p_k < -\epsilon_g \|p_k\|$, then this relaxation guarantees finite termination of the line search component in the split phase. Other related line searches employing this relaxation of the Armijo condition have been analyzed in [**8**].

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

$$f(x_k + \alpha_k^i p_k) \begin{cases} \leq f(x_k) + c_1 \alpha_k^i g(x_k)^T p_k & \text{if } i = 0,\ g(x_k)^T p_k < -\epsilon_g \|p_k\| \\ \leq f(x_k) + c_1 \alpha_k^i g(x_k)^T p_k + 2\epsilon_f & \text{if } i \geq 1,\ g(x_k)^T p_k < -\epsilon_g \|p_k\| \\ < f(x_k) & \text{if } i = 0,\ g(x_k)^T p_k \geq -\epsilon_g \|p_k\| \\ < f(x_k) + 2\epsilon_f & \text{if } i \geq 1,\ g(x_k)^T p_k \geq -\epsilon_g \|p_k\| \end{cases} \tag{3.38}$$

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

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

$$\alpha_k^{\text{best}} \in \arg\min_{\alpha_k^i} \{ f(x_k + \alpha_k^i p_k) : (3.38) \text{ is satisfied} \} \tag{3.39}$$

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

III. Initial Value of β. It is important to employ a good initial estimate of β when entering the split phase, in order to mitigate the cost of the search procedure. Recall from (3.16) that an appropriate value of the lengthening parameter is, roughly,

$$\beta_k = \frac{2(1+c_3)\epsilon_g}{m\|p_k\|_2}. \tag{3.40}$$

This formula relies on the strong convexity parameter m, which is generally not known, but since we are only using it to compute an initial value for β, it is not critical to estimate m accurately. In this vein, we compute a local estimate of m using the observed (s, y) pairs from prior iterations. For β_j with $j < k$ that satisfies both (3.34) and (3.35), we first compute an estimate of the curvature along the search direction p_j corresponding to the interval length β_j:

$$\bar{\mu}_j = \frac{(g(x_j + \beta_j p_j) - g(x_j))^T p_j}{\beta_j \|p_j\|_2^2}. \tag{3.41}$$

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

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

This aims to be only a local strong convexity estimate, whereas taking the minimum over all previous $\bar{\mu}$'s may be overly pessimistic. Let us denote by $\bar{\beta}_k$ the value obtained by making the substitution $m \leftarrow \mu_k$ in (3.40), and let β_k^i denote the ith trial lengthening parameter at iteration k, we define the initial value of the lengthening parameter for the

split phase as

$$\beta_k^{i+1} = \max\{2\beta_k^i, \bar{\beta}_k\}. \tag{3.42}$$

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

3.5. Numerical Experiments

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

In addition, we study another approach suggested by the noise control condition (3.11), based on the well known strategy of skipping a quasi-Newton update when it may not be reliable. In the BFGS (Skips) and L-BFGS (Skips) methods, the quasi-Newton update is not performed if the noise control condition is not satisfied for $c_3 = 0$, that is,

$$(g(x_k + \alpha_k p_k) - g(x_k))^T p_k < 2\epsilon_g \|p_k\|. \tag{3.43}$$

Specifically, these methods compute a steplength α_k satisfying the Armijo-Wolfe conditions (3.9)-(3.10), and if condition (3.43) holds, the BFGS update is not performed and

the next step is computed using the Hessian approximation B_k from the previous iteration; otherwise the iteration is identical to that of the BFGS and L-BFGS methods. (In the L-BFGS (Skips) method, the correction pair (s_k, y_k) is not stored when (3.43) holds.)

In summary, the 6 methods tested are:

(1) `BFGS`: the standard BFGS method given by (3.3), (3.6);

(2) `L-BFGS`: the standard L-BFGS method with memory $t = 10$; [**48**, chapter 7];

(3) `BFGS (Skips)`: the standard BFGS method given by (3.3), (3.6), but skipping the BFGS update when (3.11) is not satisfied for $\beta_k = \alpha_k$ with $c_3 = 0$;

(4) `L-BFGS (Skips)`: the standard L-BFGS method with memory $t = 10$, but skipping the L-BFGS update when (3.11) is not satisfied for $\beta_k = \alpha_k$ with $c_3 = 0$;

(5) `BFGS-E`: the noise tolerant BFGS method given by Algorithms 3.1, 3.2 and 3.3;

(6) `L-BFGS-E`: the noise tolerant L-BFGS method, which is identical to BFGS-E, except that the Hessian approximation is a limited memory matrix with memory $t = 10$.

The first four methods employ an Armijo-Wolfe line search that computes a steplength satisfying (3.9)-(3.10). The last two methods use the specialized line search described in Algorithms 3.2 and 3.3. In the deterministic case, it is common to employ cubic or quadratic interpolation to accelerate the Armijo-Wolfe search. We did not do so in the methods listed above, which use a simple bisection, because it is more robust in the presence of noise. The parameters for the line search and termination criteria are provided in Table 3.1.

We selected 41 unconstrained problems from the CUTEst collection [**31**] (see Table 3.2), and added stochastic uniform noise with different noise levels. The objective function

Table 3.1. Parameter Settings for the Methods Tested

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

and gradient have the form

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

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

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

This gives the noise bounds $|\epsilon(x)| \leq \epsilon_f = \xi_f$ and $\|e(x)\| \leq \epsilon_g = \sqrt{d}\xi_g$. Among methods for uncertainty quantification, ECNoise [**43**], point-wise sampling, and domain knowledge could be applied to obtain these bounds in practice. The optimal value ϕ^* for each function was obtained by applying the BFGS method to the original deterministic problem until it could not make further progress.

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

3.5.1. Experiments with Uniform Noise in the Gradient

In the first set of experiments, the gradient contains uniform noise but the function does not, i.e., $\epsilon_g > 0$ and $\epsilon_f = 0$. This allows us to test the efficiency of the lengthening

Table 3.2. Unconstrained CUTEst Problems Tested. d is the number of variables.

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

procedure in a benign setting that avoids the effects of the noisy line search. In these experiments, all algorithms were run for a fixed number of iterations.

We begin by revisiting the `ARWHEAD` problem from Figure 3.1, where noise was inserted with $\xi_f = 0$ and $\xi_g = 10^{-3}$. The condition number of the BFGS and BFGS-E matrices is compared in Figure 3.2, and shows that the noise control condition (3.11) stabilizes the quasi-Newton update. It may seem surprising that in Figure 3.2 the condition number of the BFGS matrix, $\kappa(H_k)$, decreases after having increased sharply. This can be explained by noting that as the iterates enter into the noisy regime, the difference in the gradient y_k can be corrupted by noise, and we may have $s_k^T y_k \gg s_k^T \tilde{y}_k$. Thus, some of the eigenvalues of the BFGS matrix $B_k = H_k^{-1}$ will increase. As the iteration proceeds, the rest of the eigenvalues become large too, hence decreasing the condition number.

Figure 3.3 plots the optimality gap $\phi(x) - \phi^*$ vs the number of gradient evaluations performed for the four methods on the `ARWHEAD` problem. BFGS and L-BFGS do not

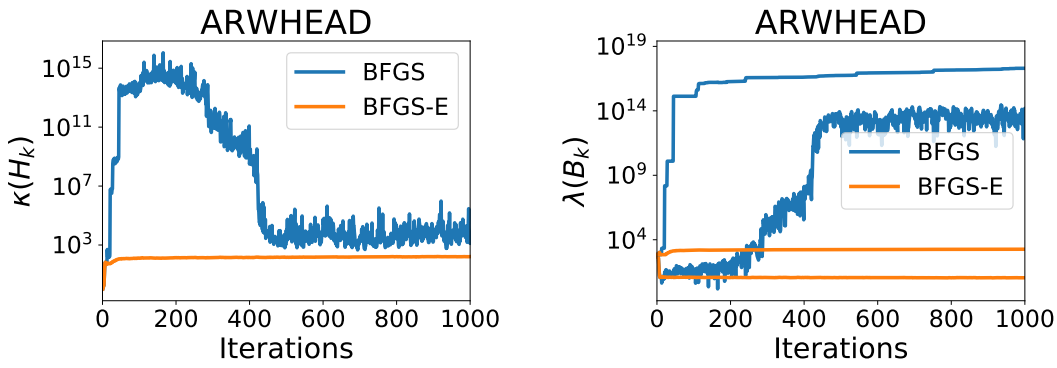

Figure 3.2. The condition number of the BFGS and BFGS-E matrices $\kappa(H_k)$ against the number of iterations (left) and the smallest and largest eigenvalues of B_k against the number of iterations (right) on the `ARWHEAD` problem. The final norm of the true gradient achieved by BFGS is approximately 1.97e−04.

achieve as high accuracy in the solution as their noise-tolerant counterparts because the deterioration in the Hessian approximation leads, at some point, to the generation of very small steps that severely limit the decrease in the objective function. The behavior of the methods on this problem is typical of what we have observed. In particular BFGS-E and L-BFGS-E trigger lengthening of the curvature pairs prior to the point where BFGS and L-BFGS stagnate due to noise. This indicates that the lengthening procedure stabilizes the Hessian approximation prior to reaching this neighborhood.

The lengthening procedure in our noise-tolerant algorithms comes at an additional computational cost. Figure 3.4 plots the cumulative number of gradient evaluations against the iteration count for the `ARWHEAD` problem. We observe that for BFGS or L-BFGS, the cumulative number of gradient evaluations is approximately equal to the number of iterations. For the noise-tolerant methods, the number of gradient evaluations match the standard BFGS and L-BFGS methods until the split phase activates. Upon

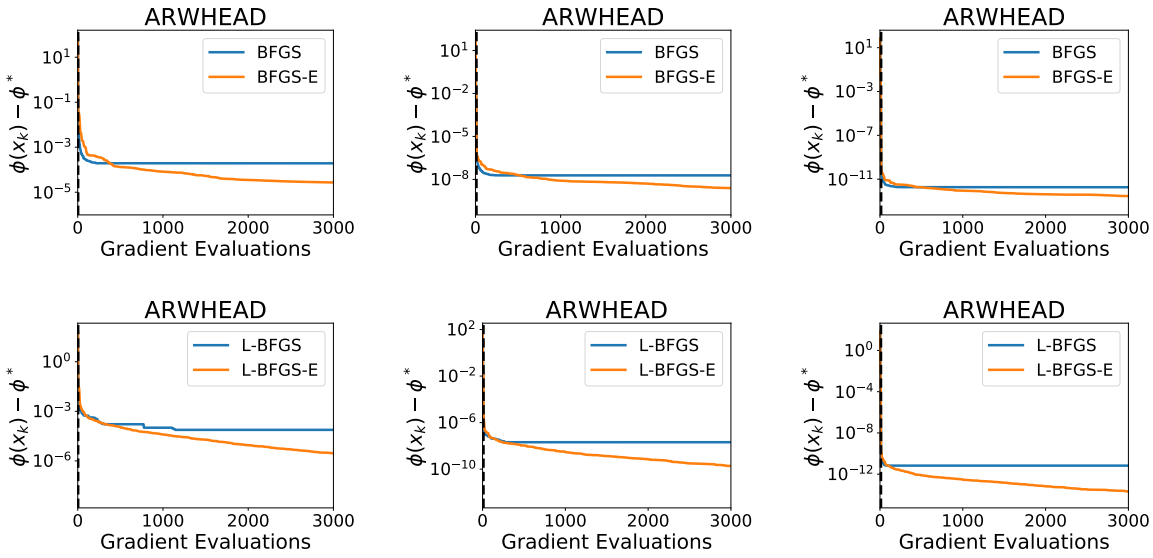

Figure 3.3. The true optimality gap $\phi(x_k) - \phi^*$ against the number of gradient evaluations on the `ARWHEAD` problem for $\epsilon_f = 0$, and for the following gradient noise levels: $\xi_g = 10^{-1}$ (left), 10^{-3} (middle), and 10^{-5} (right). The black dashed line denotes the iteration before the split phase becomes active.

entering the split phase, we notice that the cost of each iteration is approximately $2 - 4$ gradient evaluations. This may be explained by the additional $1 - 3$ gradient evaluations necessary to find the appropriate β_k that satisfies both the noise and Wolfe conditions, plus one gradient evaluation for triggering the split phase. This cost is worthwhile in that it allows the algorithm to make progress in the noisy regime.

3.5.1.1. Sensitivity with respect to ϵ_g. Since the bound on the gradient error ϵ_g may be estimated by an external procedure, it is possible for ϵ_g to be input incorrectly. In order to investigate the sensitivity of the choice of ϵ_g, we consider both under- and overestimation of it. We perform the same experiment with a fixed $\xi_g = 10^{-3}$ and $\epsilon_f = 0$

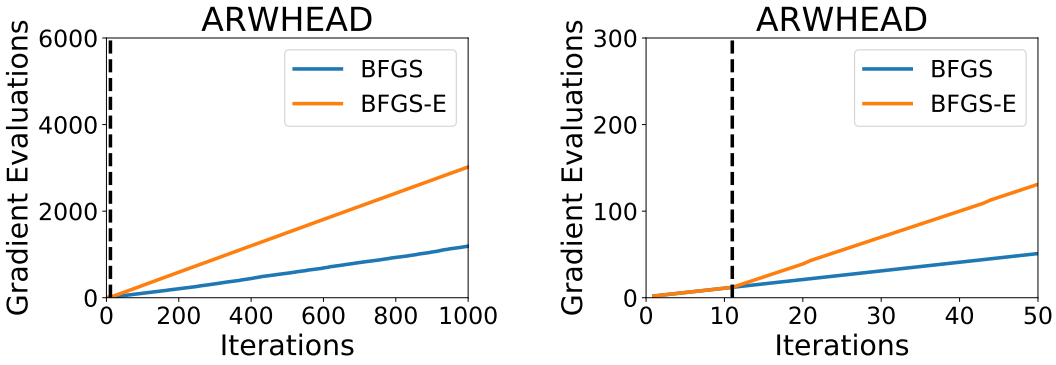

Figure 3.4. Cumulative number of gradient evaluations against the iteration count on the ARWHEAD problem for $\epsilon_f = 0$ and $\xi_g = 10^{-3}$ for BFGS and BFGS-E. The left figure plots the long-term behavior and the right figure plots the short-term behavior. The results for L-BFGS and L-BFGS-E as well as different noise levels are similar. The black dashed line denotes the iteration before the split phase becomes active.

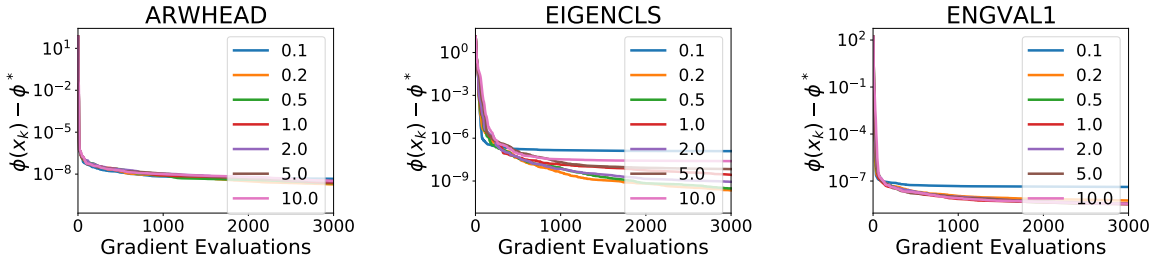

Figure 3.5. The true optimality gap $\phi(x_k) - \phi^*$ against the number of gradient evaluations applying BFGS-E on the ARWHEAD, EIGENCLS, and ENGVAL1 problems for $\epsilon_f = 0$ and $\xi_g = 10^{-3}$ with incorrectly input $\bar{\epsilon}_g = \omega\epsilon_g$ for $\omega \in \{\frac{1}{10}, \frac{1}{5}, \frac{1}{2}, 1, 2, 5, 10\}$.

but provide the algorithm an incorrect $\bar{\epsilon}_g = \omega\epsilon_g$ where $\omega \in \{\frac{1}{10}, \frac{1}{5}, \frac{1}{2}, 1, 2, 5, 10\}$. This is shown in Figure 3.5. We plot only BFGS-E since L-BFGS-E performs similarly.

If the noise is severely underestimated, it can lead to early stagnation of the algorithm due to corruption of the BFGS matrix. If the noise is severely overestimated, then the collection of non-local curvature information can result in slower progress towards the

solution. Overall, the method tolerates overestimation better than underestimation of the noise level, as one would expect.

3.5.2. Experiments with Intermittent Noise in the Gradient

In some applications, the noise level in the gradient evaluation may fluctuate rather than remain constant. One special case is that of intermittent noise. To simulate it, we will set $\xi_f = 0$ and let the noise level in the gradient ξ_g alternate between 0 and a fixed nonzero value every N_{noise} iterations, where $N_{\text{noise}} \in \{10, 25, 50\}$. We show representative results using the `CRAGGLVY` problem in Figure 3.9. The `CRAGGLVY` problem is chosen because it requires more than N_{noise} iterations to solve, whereas the `ARWHEAD` problem can be solved in under 25 iterations. The noise-tolerant methods are provided the value of ϵ_g but not N_{noise}.

BFGS suffers the most from the inclusion of intermittent noise, and is unable to recover quickly enough to make progress even when there is no noise in the gradient. In contrast, BFGS-E is able to continue to make progress immediately once noise is diminished since the BFGS matrix H_k is less corrupted by noise and therefore able to take advantage of the non-noisy gradient; see in particular the stepwise behavior on the top right plot in Figure 3.6. L-BFGS-E performs even better than BFGS-E, but note that standard L-BFGS is quite effective when the noise toggles every $N_{\text{noise}} = 25$ or 50 iterations. This is because, if the number N_{noise} of non-noisy iterations is larger than the memory $t = 10$, L-BFGS is able to forget all noise-contaminated curvature pairs, then it is able to recover and make progress.

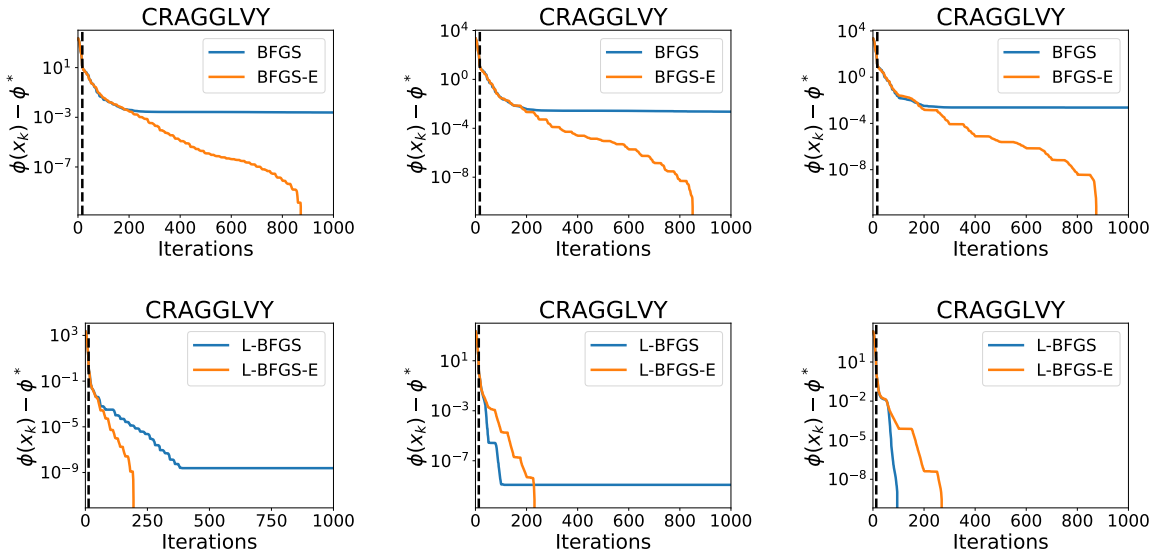

Figure 3.6. Intermittent Noise. Optimality gap $\phi(x_k) - \phi^*$ against the number of iterations on the `CRAGGLVY` problem. $\xi_f = 0$ and ξ_g alternates between 0 and with $\xi_g = 10^{-1}$ every N_{noise} iterations. Results for $N_{\text{noise}} = 10$ (left), 25 (middle), and 50 (right). The black dashed line denotes the iteration before the split phase becomes active.

3.5.3. Comparison Against Methods that Employ Update Skipping

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

In Figures 3.7 and 3.8, we compare the performance of BFGS (Skips) and L-BFGS (Skips) to both the standard and extended methods when there is uniform constant noise in the gradient. We report the results for the `ENGVAL1` and `EIGENCLS` problems, which are of easy and medium difficulty, respectively. We chose these problems to demonstrate

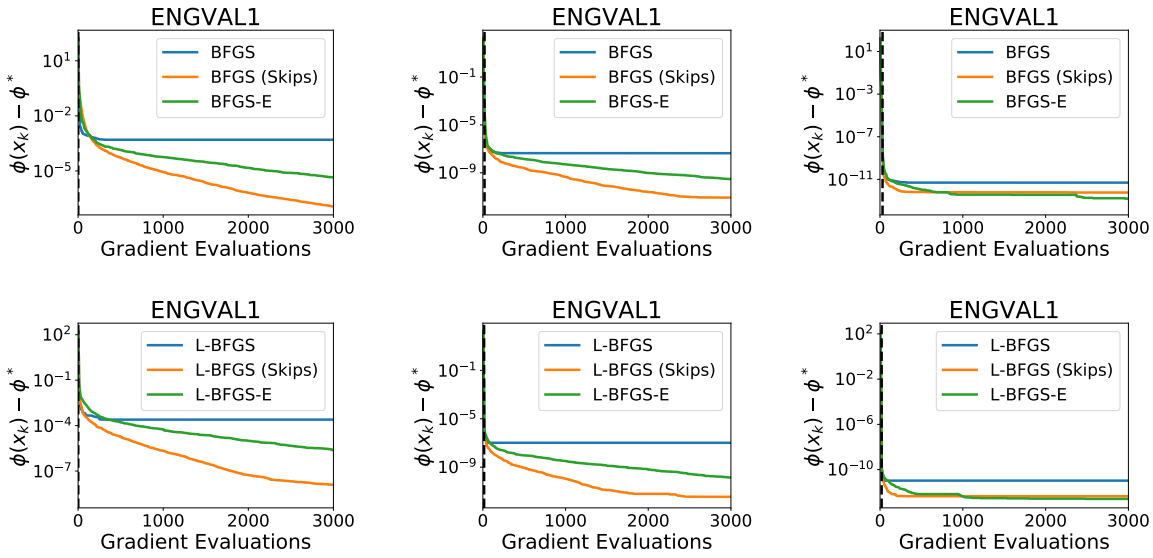

Figure 3.7. The true optimality gap $\phi(x_k) - \phi^*$ against the number of gradient evaluations on the `ENGVAL1` problem for $\epsilon_f = 0$, and for the following gradient noise levels: $\xi_g = 10^{-1}$ (left), 10^{-3} (middle), and 10^{-5} (right). The black dashed line denotes the iteration before the split phase becomes active.

nuanced cases where fixing the BFGS matrix is not sufficient for making fast progress to the solution.

In Figure 3.7, we see that BFGS (Skips) and L-BFGS (Skips) can be much more efficient than BFGS-E and L-BFGS-E. In general, we found that methods that employ update skipping can be a strong alternative to lengthening if the problem is fairly well-conditioned and the Hessian does not change much, using much fewer gradient evaluations than the two-phase line search. However, it can fail to capture the change in curvature that is necessary for more difficult problems, such as `EIGENCLS` in Figure 3.8. In such cases, continuing to update the BFGS matrix using lengthening is able to continue to

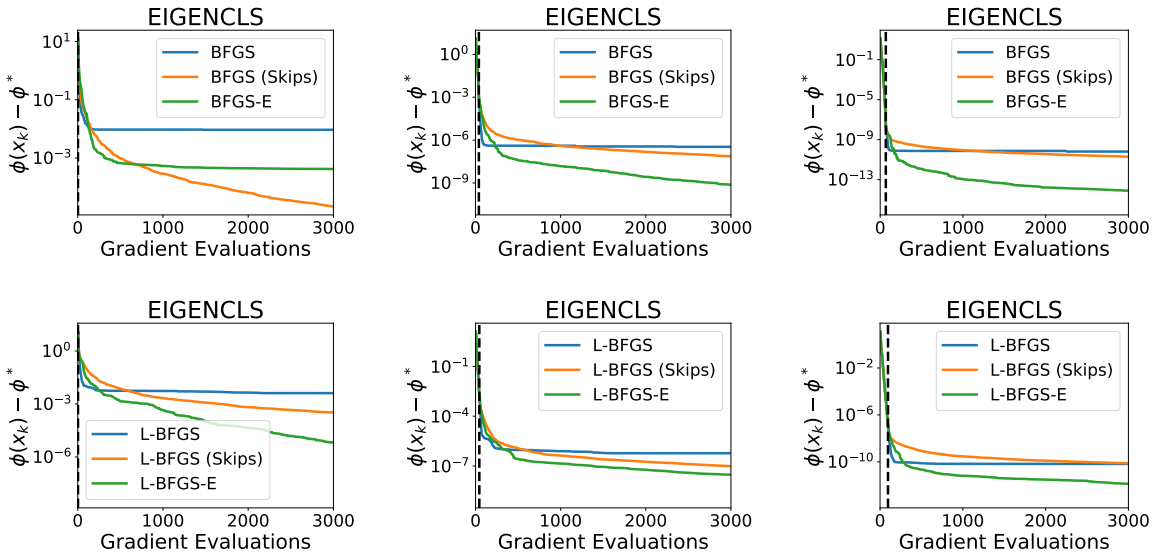

Figure 3.8. The true optimality gap $\phi(x_k) - \phi^*$ against the number of gradient evaluations on the `EIGENCLS` problem for $\epsilon_f = 0$, and for the following gradient noise levels: $\xi_g = 10^{-1}$ (left), 10^{-3} (middle), and 10^{-5} (right). The black dashed line denotes the iteration before the split phase becomes active.

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

To see how update skipping compares to lengthening in the intermittent setting, we report in Figure 3.9 results on `CRAGGLVY`, a problem of high difficulty. The skipping methods are able to make faster progress when noise is diminished but not as quickly as the noise-tolerant methods since they do not benefit from good updates to the BFGS matrix.

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

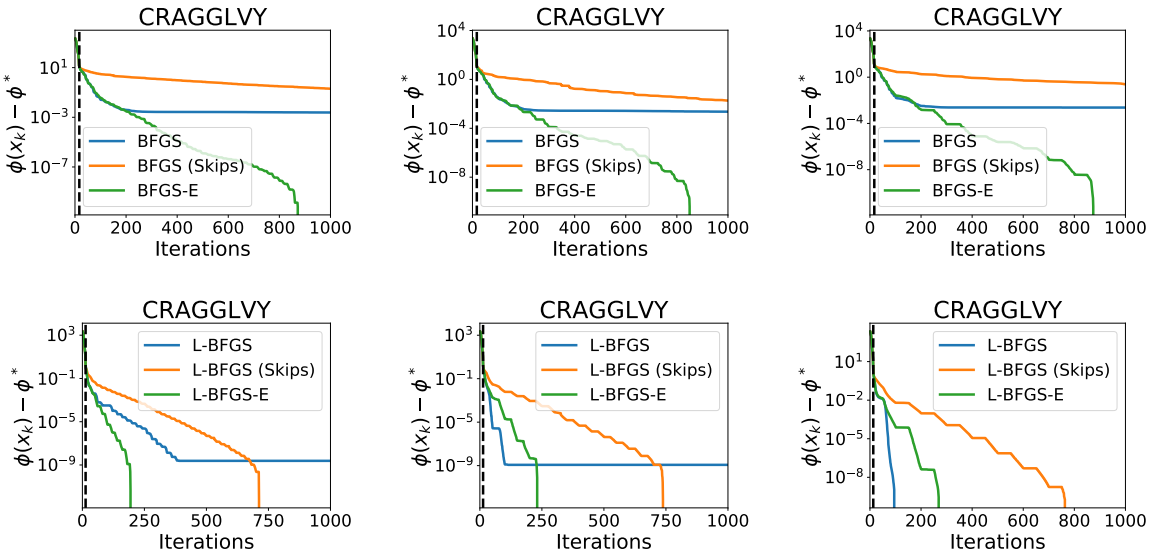

Figure 3.9. Intermittent Noise. Optimality gap $\phi(x_k) - \phi^*$ against the number of iterations on the `CRAGGLVY` problem. $\xi_f = 0$ and ξ_g alternates between 0 and with $\xi_g = 10^{-1}$ every N_{noise} iterations. Results for $N_{\text{noise}} = 10$ (left), 25 (middle), and 50 (right). The black dashed line denotes the iteration before the split phase becomes active.

3.5.4. Experiments with Function and Gradient Noise

In this set of experiments, we inject noise in both the function and gradient, i.e., $\epsilon_f, \epsilon_g > 0$. First, we report in Figures 3.10 and 3.11 results for a representative example: problem `DIXMAANH`. We ran all methods for 3000 gradient evaluations to illustrate their long term behavior, for different values of ϵ_g and ϵ_f. We note that the lengthening procedure safeguards the Hessian updating in the presence of function noise, and the relaxation in the Armijo condition (3.38) allows the methods to continue making progress far below the noise level of the function if the gradient noise is sufficiently small to provide good search directions.

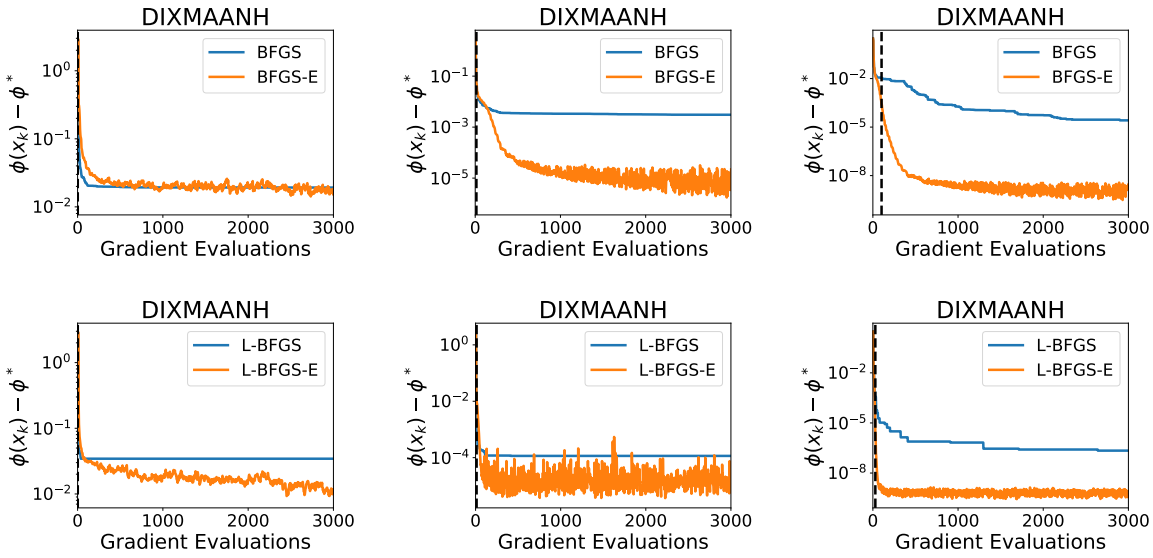

Figure 3.10. Optimality gap $\phi(x_k) - \phi^*$ against the number of gradient evaluations on problem `DIXMAANH`, with $\xi_f = 10^{-3}$ on all six plots, and with $\xi_g = 10^{-1}$ (left), $\xi_g = 10^{-3}$ (middle), and $\xi_g = 10^{-5}$ (right). The black dashed line denotes the iteration before the split phase becomes active.

Lastly, we report the performance of the methods on the 41 test problems listed in Table 2, using the profiles proposed by Morales [**42**]. In Figures 3.12 and 3.13 we report, respectively, the quantities

$$\log_2\left(\frac{\phi_{new} - \phi^*}{\phi_{old} - \phi^*}\right) \text{ and } \log_2\left(\frac{evals_{new}}{evals_{old}}\right), \tag{3.44}$$

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

$$\phi(x_k) - \phi^* \leq \epsilon_f \text{ or } \|\nabla\phi(x_k)\| \leq \epsilon_g. \tag{3.45}$$

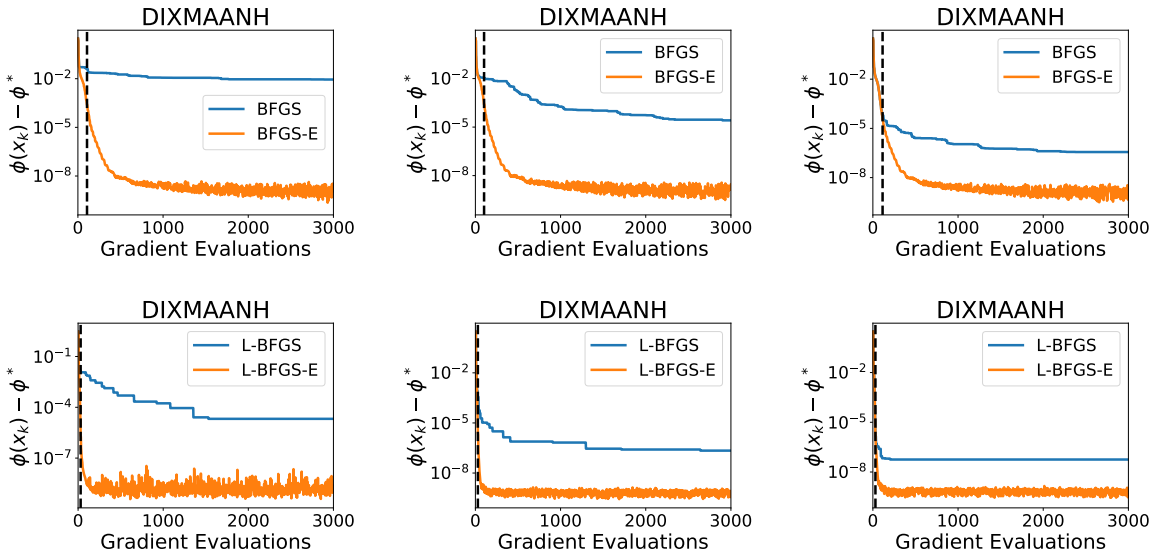

Figure 3.11. Optimality gap $\phi(x_k) - \phi^*$ against the number of gradient evaluations on problem `DIXMAANH` with $\xi_g = 10^{-5}$ on all six plots, and with $\xi_f = 10^{-1}$ (left), $\xi_f = 10^{-3}$ (middle), and $\xi_f = 10^{-5}$ (right). The black dashed line denotes the iteration before the split phase becomes active.

All quantities are averaged over 5 runs with different seeds. In Figures 3.12 and 3.13 the problems are ordered in increasing value of the quantities given in (3.44). One can thus gauge the success of a method by the area of the graph on its side of the half-space: the larger the area, the more successful the method.

Figure 3.12 thus compares the (long term) ability of the methods to achieve high accuracy in the function value, whereas Figure 3.13 measures the short-term cost in terms of gradient evaluations to achieve the noise level in the function or gradient. These results suggest that the noise tolerant methods often provide a real improvement in the solution of certain classes of optimization problems with noisy function and gradient evaluations.

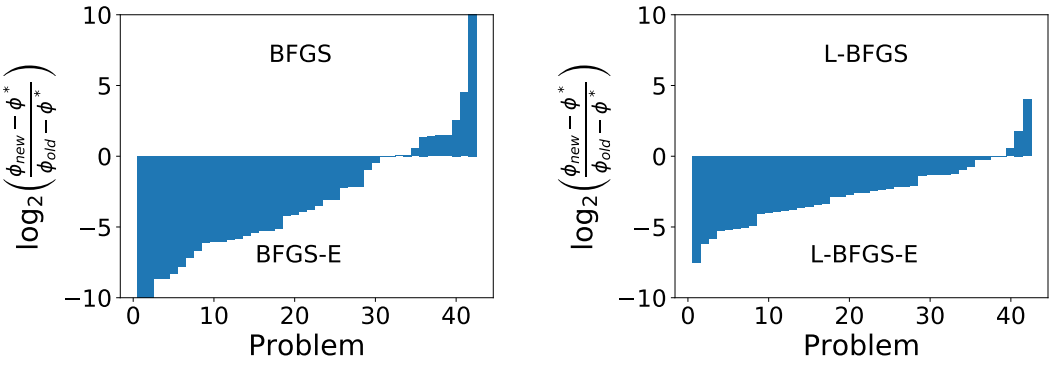

Figure 3.12. Morales profiles for the optimality gap $\phi(x_k) - \phi^*$ across 41 unconstrained CUTEst problems with $\xi_f = 10^{-3}$ and $\xi_g = 10^{-3}$. Results are averaged over 5 runs. The left figure compares BFGS against BFGS-E while the right figure compares L-BFGS against L-BFGS-E.

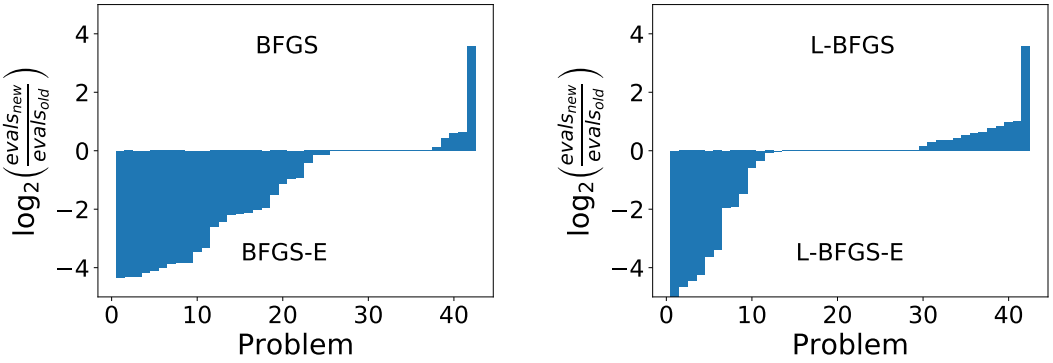

Figure 3.13. Morales profiles for the total number of gradient evaluations to achieve (3.45) across 41 unconstrained CUTEst problems with $\xi_f = 10^{-3}$ and $\xi_g = 10^{-3}$. Results are averaged over 5 runs. The left figure compares BFGS against BFGS-E while the right figure compares L-BFGS against L-BFGS-E.

3.6. Final Remarks

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

This chapter makes two contributions. It introduces the noise control condition (3.11), which can be used to determine when to skip a quasi-Newton update or adaptively lengthen the interval from which gradient differences can be employed reliably. Our proposed BFGS-E and L-BFGS-E methods utilize the latter and enjoy convergence guarantees to a neighborhood of the solution for strongly convex functions.

The second contribution of the chapter is to show that the lengthening procedure based on condition (3.11) is successful in practice, and thus transforms the theoretical algorithm proposed in [**58**] into a robust and practical procedure. Our numerical experiments show that quasi-Newton updating remains stable after the algorithm has reached the region where errors dominate, and this allows the noise tolerant methods to reach higher accuracy in the solution. Our testing also shows that the proposed algorithms are not more expensive than the standard BFGS and L-BFGS methods in the region where the latter two methods operate reliably. Once the iterates reach a neighborhood where BFGS updating is corrupted and the iteration stalls, the new algorithms invoke the lengthening procedure that typically requires $2-4$ gradient evaluations per iteration. We also tested an update skipping strategy based on the noise tolerant condition. We found that, although update skipping can be very efficient when applied to easy problems with uniform noise, the noise tolerant methods are more efficient when applied to harder problems or problems with oscillating noise.

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

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

CHAPTER 4

Adaptive Finite-Difference Estimation

4.1. Introduction

The problem of interest is the unconstrained minimization of a smooth function $\phi : \mathbb{R}^n \to \mathbb{R}$

$$\min_{x \in \mathbb{R}^n} \phi(x) \tag{4.1}$$

while only given access to noisy evaluations of the objective function $f(x) = \phi(x) + \epsilon(x)$ where the function $\epsilon : \mathbb{R}^n \to \mathbb{R}$ models the error. We assume that the noise is bounded, i.e., for $\epsilon_f \geq 0$, $|\epsilon(x)| \leq \epsilon_f$ for all $x \in \mathbb{R}^n$. We will call ϵ_f the *noise level* of f. The noise level is assumed to be known to the user, either *a priori* or estimated via sampling or difference tables; see, for example, [**43**].

In this work, we consider the finite-difference methods, where a gradient approximation $g(x; h)$ is computed by

$$[g(x; h)]_i = \frac{f(x + he_i) - f(x)}{h} \tag{4.2}$$

where h is the finite-difference interval and is typically chosen as

$$h = \max\{1, \sqrt{|x_i|}\}\sqrt{\epsilon_m},$$

where ϵ_m denotes machine precision. Note that $n+1$ function evaluations are needed to evaluate $g(x;h)$. Although this approach is sensible in the noiseless setting, this particular choice of the finite-difference interval h may be poor under the presence of large errors. Consider the following decomposition of the forward-difference formula:

$$[g(x;h)]_i = \frac{\phi(x+he_i)-\phi(x)}{h} + \frac{\epsilon(x+he_i)-e(x)}{h}.$$

The error induced by the first term is called *truncation error* due to error resulting from truncation of the Taylor series, and the error induced by the second term is called *measurement error*.

This decomposition explains the potential issues that arise with the choice of h: if h is chosen too small, then measurement error from inexact function evaluations will dominate the approximation, while if h is chosen too large, then the approximation may remain poor due to truncation error. The ideal choice of the finite-difference interval must balance both sources of error. The sensitivity of this choice of the finite-difference interval has prevented finite-difference methods from being used more widely in derivative-free optimization, as black-box problems are frequently coupled with noise.

Theoretically, one can bound the errors that arise from each of these terms and derive an optimal choice of the finite-difference interval h that minimizes this upper bound. However, the finite-difference formulas rely on knowledge of the noise level ϵ_f as well as bounds on high-order derivatives of the objective function in a neighborhood around x. While estimating the noise level can be done using `ECnoise` or pointwise sample variance estimates, cheaply and accurately determining bounds on high-order derivatives is far more difficult. Various procedures have been developed in the past for determining L_2,

the bound on the second order derivative; see [**28, 44**]. These approaches attempt to select an appropriate finite-difference interval for approximating L_2, then employs this within the ideal finite-difference interval formula. These heuristics, however, suffer from multiple weaknesses: (1) some are not affine-invariant, that is, applying the heuristic to a modified evaluation of the function $a \cdot f(x) + b$ for some $a, b \in \mathbb{R}$ $(a \neq 0)$ yields different behavior of the algorithm; (2) the methods are restricted to the estimation of L_2 for forward differences; (3) the methods can suffer from symmetry within the function; (4) they lack rigorous theoretical treatment; and (5) they are not integrated within an optimization algorithm, that is, they are typically only employed at the beginning of each run. This approach has been demonstrated to result in low-accuracy solutions, as noted by [**2, 5, 57**].

We propose a generalized adaptive procedure for obtaining near-optimal estimates of the finite-difference interval for any finite-difference scheme for approximating the d-th order derivative of a noisy univariate function. The approach relies on a bisection search over the finite-difference interval to balance the truncation and measurement error. While the approach may be viewed as a generalization of Gill, et al. [**28**], we develop and analyze an alternative interpretation of the approach.

This chapter is organized into six sections. In Section 2, we introduce our finite-difference interval estimation procedure. In Section 3, we provide guarantees for termination of our procedure, as well as guarantees that the output differencing interval is optimal up to a constant. In Section 4, we present numerical experiments demonstrating practical performance of our proposed approach, and concluding remarks are made in Section 5.

4.2. Adaptive Finite-Difference Interval Estimation

In this section, we will consider the estimation of the finite-difference interval for a noisy univariate function for computing an accurate finite-difference approximation of its derivative. We will let $v : \mathbb{R} \to \mathbb{R}$ denote a univariate noisy function which satisfies

$$v(t) = \hat{v}(t) + \epsilon(t),$$

where $\hat{v}(t)$ is the underlying smooth function and $\epsilon(t)$ is the noise in the function that satisfies $|\epsilon(t)| \leq \epsilon_f$ for all $t \in \mathbb{R}$. Over the course of this section, we will generally use $\hat{v}^{(d)}(t)$ to denote the true d-th derivative of $\hat{v}$. One can extend this procedure to the multivariate case by applying the procedure to each component variable, i.e., defining $v(t) = f(x_0 + te_i)$ and $\hat{v}(t) = \phi(x_0 + te_i)$ for some reference point $x_0 \in \mathbb{R}^n$ and some $i \in \{1, ..., n\}$. We will first consider the simple forward-difference case in Section 4.2.1, then generalize our approach to arbitrary finite-difference schemes in Section 4.2.2.

4.2.1. Forward-Difference Interval Estimation of Univariate Functions

Suppose we are interested in determining the finite-difference interval for the forward-difference approximation of the first derivative of $\hat{v}$. In particular, let

$$v^{(1)}(t; h) = \frac{v(t + h) - v(t)}{h} \approx \hat{v}^{(1)}(t) \tag{4.3}$$

denote the forward-difference approximation to $\hat{v}^{(1)}(t)$. Since the Taylor expansion of the true function $\hat{v}$ is given by

$$\hat{v}(t + h) = \hat{v}(t) + \hat{v}^{(1)}(t)h + \frac{\hat{v}^{(2)}(t)}{2}h^2 + o(h^2),$$

the total error can be bounded by

$$|\hat{v}^{(1)}(t)-v^{(1)}(t)| = \left| \left(\frac{\hat{v}(t+h)-\hat{v}(t)}{h} - \hat{v}^{(1)}(t) \right) + \frac{\epsilon(t+h)-\epsilon(t)}{h} \right| \leq \frac{|\hat{v}^{(2)}(t)|h}{2} + \frac{2\epsilon_f}{h} + o(h).$$

Ignoring the higher-order term, this yields an optimal interval (with respect to the upper bound) of

$$h^* \approx 2\sqrt{\frac{\epsilon_f}{|\hat{v}^{(2)}(t)|}}. \tag{4.4}$$

Therefore, in order to attain a near-optimal interval, we want to design a procedure that will yield an interval $h = O\left(\sqrt{\epsilon_f/|\hat{v}^{(2)}(t)|}\right)$.

To do this, we will construct a *testing ratio* which we define as

$$r(h; v, t, \epsilon_f) = \frac{|v(t+2h) - 2v(t+h) + v(t)|}{4\epsilon_f}. \tag{4.5}$$

Given $r_l > 1$ and $r_u > r_l + 2$, we will perform a bisection search to find an interval $h > 0$ that satisfies

$$r(h; v, t, \epsilon_f) \in [r_l, r_u]. \tag{4.6}$$

In particular, if $r(h; v, t, \epsilon_f) < r_l$, then the numerator is dominated by noise, indicating that h is too small. On the other hand, if $r(h; v, t, \epsilon_f) > r_u$, then the numerator significantly dominates the noise, which implies that h is too large. Note that if one monotonically increases or decreases h by a factor of 2, the new trial h only requires a single new function evaluation to evaluate the testing ratio.

To see why this procedure works to give us a near-optimal h, note that

$$\hat{v}(t+2h) - 2\hat{v}(t+h) + \hat{v}(t) = \hat{v}^{(2)}(t)h^2 + o(h^2). \tag{4.7}$$

Therefore, if we expand the testing ratio, we obtain:

$$r(h; v, t, \epsilon_f) = \left| \frac{\hat{v}^{(2)}(t)h^2}{4\epsilon_f} + \frac{\epsilon(t+2h) - 2\epsilon(t+h) + \epsilon(t)}{4\epsilon_f} + o(h^2) \right|. \tag{4.8}$$

By (4.6) and ignoring the $o(h^2)$ term, since $\left|\frac{\epsilon(t+2h)-2\epsilon(t+h)+\epsilon(t)|}{4\epsilon_f}\right| \leq 1$ by the fact that $|\epsilon(t)| \leq \epsilon_f$ for all $t \in \mathbb{R}$, we approximately have

$$r(h; v, t, \epsilon_f) \in [r_l, r_u] \Rightarrow h \in 2\sqrt{\frac{\epsilon_f}{|\hat{v}^{(2)}(t)|}} \cdot [\sqrt{r_l - 1}, \sqrt{r_u + 1}]. \tag{4.9}$$

Therefore, if r_l and r_u are chosen properly, we obtain $h = O\left(\sqrt{\frac{\epsilon_f}{|\hat{v}^{(2)}(t)|}}\right)$, which is the same order as the optimal finite-difference interval (4.4).

Remark 1. Although the definition of the testing ratio is similar to Gill, et al. [**28**], our approach markedly differs from theirs in three aspects: (1) we utilize the h derived from the testing ratio directly as the chosen finite-difference interval, whereas Gill, et al. uses this to estimate the second derivative; (2) our approach utilizes a bisection search to find h rather than monotonically increasing or decreasing h; and (3) it relies on forward-difference instead of central-difference estimates.

The first aspect follows from the observation made in (4.9) that $h = O\left(\sqrt{\frac{\epsilon_f}{|\hat{v}^{(2)}(t)|}}\right)$. Hence, the finite-difference interval h obtained by our bisection procedure is near-optimal in that it is only a constant factor away from the optimal interval. Two observations can

be made from this derivation. The first is that since $v(t+h)$ and $v(t)$ are used in the evaluation of the testing ratio, one can reuse prior function evaluations from the bisection procedure to estimate $\hat{v}^{(1)}(t)$. The second observation is that the derivation motivates an initial choice of $h = O(\sqrt{\epsilon_f})$ as opposed to $h = O(\sqrt[4]{\epsilon_f})$, as used in Moré and Wild [**44**].

The second aspect follows from different tradeoffs between cost and accuracy in the finite-difference interval estimate. In particular, whereas the Gill, et al. procedure is cheaper but yields a less accurate estimate, our procedure is able to guarantee a sufficiently accurate estimate at higher cost.

Lastly, the third aspect was designed to avoid cancellation due to symmetry in the numerator of the testing ratio. In particular, Gill, et al. [**28**] rely on a testing ratio using the central difference:

$$\frac{|v(t+h) - 2v(t) + v(t-h)|}{4\epsilon_f} \in [r_l, r_u]. \tag{4.10}$$

However, we can obtain poor estimates of the derivative when the function has near central symmetry at the point of interest, for example, on function $\hat{v}(t) = t^3 + t$ near $t = 0$ with sufficiently large noise.

Remark 2. Note that the testing ratio is affine-invariant in the sense that $r(h; v, t, \epsilon_f)$ remains unchanged if applied to a modified function $\bar{v}(t) = a \cdot v(t) + b$ for $a \neq 0$ and $b \in \mathbb{R}$ with noise level $|a|\epsilon_f$, i.e. $r(h; \tilde{v}, t, |a|\epsilon_f) = r(h; v, t, \epsilon_f)$. Therefore, the finite-difference interval h will correctly remain unchanged under this transformation. This differs from

Moré and Wild's procedure [**44**] which relies on the condition

$$|v(t \pm h) - v(t)| \leq \tau_2 \max\{|v(t)|, |v(t \pm h)|\}$$

with $\tau_2 \in (0, 1)$ to ensure that h is not too large. This condition is not affine-invariant in the sense that adding a sufficiently large b can force the condition to be satisfied.

4.2.2. Generalized Finite-Difference Interval Estimation of Univariate Functions

To our knowledge, all previous finite-difference interval estimation procedures for numerical optimization solely focused on forward differences [**2, 28, 44**]. However, it has been found, particularly in the noisy regime, that higher-order finite-difference approximations, such as central differences, can yield solutions with higher accuracy; see [**57**]. To handle these cases, we generalize our finite-difference interval estimation scheme to generic finite-difference approximations for d-th order derivatives.

Consider a general finite-difference approximation scheme $S = (w, s)$ defined over p points, where we approximate $\hat{v}^{(d)}(t)$ using the equation

$$v_S^{(d)}(t; h) = \frac{\sum_{j=1}^{p} w_j \cdot v(t + hs_j)}{h^d} \approx \hat{v}^{(d)}(t) \tag{4.11}$$

where $w \in \mathbb{R}^p$ and $s \in \mathbb{R}^p$ are the associated weights and shifts of the finite-difference scheme. As in the forward-difference setting, we denote the finite-difference approximation as $v_S^{(d)}(t; h)$. Common examples include *forward-difference* and *central-difference* schemes for approximating the first derivative (i.e., $d = 1$), where s and w are defined as $s = (0, 1)^T$ and $w = (-1, 1)^T$ and $s = (-1, 1)^T$ and $w = (-\frac{1}{2}, \frac{1}{2})^T$, respectively.

In order for the finite-difference scheme to be valid, the finite-difference coefficients w and shifts s must be chosen such that the Taylor expansion of the finite-difference approximation over the true function satisfies

$$\sum_{j=1}^{p} w_j \cdot \hat{v}(t + hs_j) = \hat{v}^{(d)}(t)h^d + c_q\hat{v}^{(q)}(t)h^q + o\left(h^q\right). \tag{4.12}$$

where $c_q \neq 0$, and $q \geq d+1$ denotes the order of the remainder term. In order to guarantee this, the finite-difference scheme S must satisfy

$$\frac{1}{l!}\sum_{j=1}^{p} w_j s_j^l = \begin{cases} 1 & \text{if } l = d \\ 0 & \text{if } 0 \leq l < q,\ l \neq d \end{cases}$$

and

$$c_q = \frac{1}{q!}\sum_{j=1}^{p} w_j s_j^q \neq 0.$$

Therefore, in the presence of noise, the worst-case error for the finite-difference scheme of interest can be bounded by

$$|v^{(d)}(t) - \hat{v}^{(d)}(t)| \leq |c_q| \left|\hat{v}^{(q)}(t)\right| h^{q-d} + \|w\|_1 \epsilon_f h^{-d} + o\left(h^{q-d}\right).$$

Assuming knowledge of these constants, one can define an (approximately) optimal choice of h:

$$h^* \approx \left|\frac{d}{q-d} \cdot \frac{\|w\|_1 \epsilon_f}{c_q \hat{v}^{(q)}(t)}\right|^{1/q} = O\left(\left|\frac{\epsilon_f}{\hat{v}^{(q)}(t)}\right|^{1/q}\right). \tag{4.13}$$

While ϵ_f is assumed to be known and d, q, w and c_q are readily available or are easily derived, the q-th order derivative $\hat{v}^{(q)}(x)$ is unknown and often difficult to compute or

estimate. Following the idea from the forward-difference case, we will first construct a testing ratio r_S associated with scheme S:

$$r_S(h; v, t, \epsilon_f) = \frac{\left|\sum_{j=1}^{\tilde{p}} \tilde{w}_j \cdot v(t + h\tilde{s}_j)\right|}{\epsilon_f}$$

where $\tilde{w}, \tilde{s} \in \mathbb{R}^{\tilde{p}}$ satisfy

$$\sum_{j=1}^{\tilde{p}} \tilde{w}_j \cdot \hat{v}(t + h\tilde{s}_j) = c_t \hat{v}^{(q)}(t) h^q + o\left(h^q\right), \; c_t = \frac{1}{q!} \sum_{j=0}^{q} \tilde{w}_j \tilde{s}_j^q \neq 0. \tag{4.14}$$

Without loss of generality, we require $\tilde{w}$ to satisfy

$$\|\tilde{w}\|_1 = 1,$$

the reason for which will be evident below. We will then perform a bisection search to find an interval h that satisfies

$$r_S(h; v, t, \epsilon_f) \in [r_l, r_u] \tag{4.15}$$

for some $r_l > 1$ and $r_u > r_l + 2$. The procedure is summarized in Algorithm 4.1.

Algorithm 4.1: *Adaptive Finite-Difference Interval Estimation*

Input: One-dimensional noisy function $v : \mathbb{R} \to \mathbb{R}$; noise level $\epsilon_f > 0$; testing ratio $r_S(h; v, t, \epsilon_f)$ for scheme S; lower- and upper-bound: r_l and r_u satisfying $0 < r_l < r_u$; initial scaling factor h_0
$h \leftarrow h_0$
$l \leftarrow 0$, $u \leftarrow +\infty$
while *true* **do**
    Evaluate $r_S(h; v, t, \epsilon_f)$
    if $r_S(h; v, t, \epsilon_f) < r_l$ **then**
        $l \leftarrow h$
    else if $r_S(h; v, t, \epsilon_f) > r_u$ **then**
        $u \leftarrow h$
    else
        break
    end if
    if $u = +\infty$ **then**
        $h \leftarrow 2l$
    else
        $h \leftarrow (l + u)/2$
    end if
end while
return h

By (4.14), we can see that

$$r_S(h; v, t, \epsilon_f) = \left| \frac{c_t \hat{v}^{(q)}(t) h^q}{\epsilon_f} + \frac{\sum_{j=1}^{\tilde{p}} \tilde{w}_j \cdot \epsilon(t + h\tilde{s}_j)}{\epsilon_f} + o\left(h^q\right) \right| \tag{4.16}$$

$$= \left| \frac{c_t \hat{v}^{(q)}(t) h^q}{\epsilon_f} + \Delta + o\left(h^q\right) \right|, \ |\Delta| \leq 1 \tag{4.17}$$

where we have defined

$$\Delta = \frac{\sum_{j=1}^{\tilde{p}} \tilde{w}_j \cdot \epsilon(t + h\tilde{s}_j)}{\epsilon_f}$$

to simplify the notation. The fact that $|\Delta| \leq 1$ is a consequence of the requirement that $\|\tilde{w}\|_1 = 1$ and that $|\epsilon(t)| \leq \epsilon_f$. Therefore, if we have $r_S(h; v, t, \epsilon_f) \in [r_l, r_u]$ and if we ignore $o(h^q)$ term, then we (approximately) have

$$\left| \frac{c_t \hat{v}^{(q)}(t) h^q}{\epsilon_f} \right| \in [r_l - 1, r_u + 1] \tag{4.18}$$

i.e.,

$$h \in \left[\left(\frac{r_l - 1}{|c_t|} \frac{\epsilon_f}{|\hat{v}^{(q)}(t)|} \right)^{1/q}, \left(\frac{r_u + 1}{|c_t|} \frac{\epsilon_f}{|\hat{v}^{(q)}(t)|} \right)^{1/q} \right]. \tag{4.19}$$

If we look at (4.13), we can see that h has the same dependence on ϵ_f and $\hat{v}^{(q)}(t)$ as h^*. Then, with a suitable choice of the constants r_l, r_u, we can ensure that the h output by Algorithm 4.1 is nearly optimal.

To see how this works, consider the examples for forward differencing and central differencing below.

Example 1 (Forward Difference). Consider the forward-difference scheme S for approximating the first derivative:

$$v_S^{(1)}(t) = \frac{v(t + h) - v(t)}{h} \tag{4.20}$$

where $s = (0, 1)^T$ and $w = (-1, 1)^T$. Note that the Taylor expansion over the true function is given as:

$$\frac{\hat{v}(t+h) - \hat{v}(t)}{h} = \hat{v}^{(1)}(t) + \frac{\hat{v}^{(2)}(t)h}{2} + o(h).$$

Therefore, the full error of the derivative approximation and the approximate optimal choice of h is:

$$|v_S^{(1)}(t) - \hat{v}^{(1)}(t)| \leq \frac{|\hat{v}^{(2)}(t)|h}{2} + \frac{2\epsilon_f}{h} + o(h),\ h^* \approx 2\sqrt{\frac{\epsilon_f}{|v^{(2)}(t)|}}. \tag{4.21}$$

One example of a valid testing ratio is:

$$r_S(h; v, t, \epsilon_f) = \frac{|v(t+2h) - 2v(t+h) + v(t)|}{4\epsilon_f}. \tag{4.22}$$

Example 2 (Central Difference). Consider the central-difference scheme for approximating the first derivative:

$$v_S^{(1)}(t) = \frac{v(t+h) - v(t-h)}{2h} \tag{4.23}$$

where $s = (-1, 1)^T$ and $w = (-\frac{1}{2}, \frac{1}{2})^T$. The Taylor expansion of the numerator is given as:

$$\frac{\hat{v}(t+h) - \hat{v}(t-h)}{2h} = \hat{v}^{(1)}(t) + \frac{\hat{v}^{(3)}(t)h^2}{6} + o(h^2).$$

The full error of the derivative approximation and the approximate optimal choice of h is:

$$\left|v_S^{(1)}(t) - \hat{v}^{(1)}(t)\right| \leq \frac{\left|\hat{v}^{(3)}(t)\right| h^2}{6} + \frac{\epsilon_f}{h} + o(h^2),\ h^* \approx \sqrt[3]{\frac{3\epsilon_f}{|\hat{v}^{(3)}(t)|}}.$$

One example of a valid testing ratio is:

$$r_S(h; v, t, \epsilon_f) = \frac{|v(t+2h) - 2v(t+h) + 2v(t-h) - v(t-2h)|}{6\epsilon_f}. \tag{4.24}$$

To our knowledge, our derivation and analysis are the first for procedures of this kind. To summarize, we present examples of testing ratios $r_S(h; v, t, \epsilon_f)$ for commonly used finite-difference schemes in Table 4.1. As in the forward-difference case, our procedure is invariant to affine transformations.

finite-difference approximation	testing ratio $r_S(h; v, t, \epsilon_f)$	Taylor leading term
$\hat{v}^{(1)}(t) \approx \frac{v(t+h)-v(t)}{h}$	$\frac{\lvert v(t+2h)-2v(t+h)+v(t)\rvert}{4\epsilon_f}$	$\frac{\lvert\hat{v}^{(2)}(t)\rvert h^2}{4\epsilon_f}$
$\hat{v}^{(1)}(t) \approx \frac{v(t+h)-v(t-h)}{2h}$	$\frac{\lvert v(t+2h)-2v(t+h)+2v(t-h)-v(t-2h)\rvert}{6\epsilon_f}$	$\frac{\lvert\hat{v}^{(3)}(t)\rvert h^3}{3\epsilon_f}$
$\hat{v}^{(2)}(t) \approx \frac{v(t+h)-2v(t)+v(t-h)}{h^2}$	$\frac{\lvert v(t+2h)-4v(t+h)+6v(t)-4v(t-h)+v(t-2h)\rvert}{16\epsilon_f}$	$\frac{\lvert\hat{v}^{(4)}(t)\rvert h^4}{16\epsilon_f}$

Table 4.1. Commonly used finite-difference schemes, their testing ratios using the doubling trick, and their leading terms in the Taylor expansion.

4.2.2.1. Practical Considerations. In order to make this procedure practical, we discuss a number of considerations below.

I. Generation of Testing Ratio. Although many choices of r_S are possible for a finite-difference scheme S, it would be ideal to be able to automatically generate a testing ratio given any finite-difference scheme S. A simple yet useful way to construct r_S is through the formula

$$r_S^{\alpha}(h; v, t, \epsilon_f) = \frac{\left|v_S^{(d)}(t; h) - v_S^{(d)}(t; \alpha h)\right|}{A\epsilon_f / h^d}, \tag{4.25}$$

where $\alpha \neq 1$, and A is computed by normalizing the coefficients such that $\|\tilde{w}\|_1 = 1$ is satisfied. For any $\alpha \neq 1$, this yields a valid testing ratio r_S. A common choice of α is $\alpha = 2$, which we will refer to as the *doubling trick*.

This approach is guaranteed to generate a valid r_S because it cancels out the $\hat{v}^{(d)}(t)$ term in the Taylor expansion, leaving only the relevant higher-order term of order q of interest. The doubling trick allows us to reuse some function evaluations at each iteration. In addition, since the function evaluations needed for the finite-difference approximation are already computed within our adaptive procedure, one can immediately evaluate the finite-difference approximation with the saved function values without any additional function evaluations.

II. Choice of r_l and r_u. Ideally, one should choose r_l and r_u such that they are close to

$$\frac{d}{q-d} \cdot \left|\frac{c_t}{c_q}\right| \cdot \|w\|_1$$

in order to yield an h that is close to h^* in (4.13). Unfortunately, this is not possible due to the presence of noise, which requires that $1 < r_l < r_u - 2$ in order to ensure finite-termination; see Section 4.2.2.2. We therefore set

$$r_l = \max\left\{1.1,\ \frac{1}{2} \cdot \frac{d}{q-d} \cdot \left|\frac{c_t}{c_q}\right| \cdot \|w\|_1\right\},\ r_u = 3r_l. \tag{4.26}$$

Note that this may result in the overestimation of h. However, this has not been overly problematic in practice as it typically differs by a constant factor.

III. Initialization of h_0. Noting that the h that satisfies the procedure is approximately of the form (4.19), it is preferable to initialize $h_0 = \epsilon_f^{1/q}$. If the finite-difference interval is reestimated within an optimization algorithm, we can initialize h_0 as the h used at the prior iteration.

IV. Handling of Special Cases. The Taylor expansion analysis elucidates two possible failure cases for our procedure. In particular, observe that

$$r_S(h; v, t, \epsilon_f) = \left| \frac{c_t \widehat{v}^{(q)}(t) h^q}{\epsilon_f} + \Delta + O\left(\frac{\widehat{v}^{(q+1)}(\xi) h^{q+1}}{\epsilon_f} \right) \right|$$

where $\xi \in [\min_j\{t + h\tilde{s}_j\}, \max_j\{t + h\tilde{s}_j\}]$. If $\widehat{v}^{(q+1)}(\xi)$ is large, then the higher-order term can dominate the other terms. This can produce poor estimates of h, particularly when ϵ_f and h^* is large, even if the condition $r_S(h; v, t, \epsilon_f) \in [r_l, r_u]$ is satisfied.

The other special case occurs if $\widehat{v}^{(q)}(\xi) = 0$ or $\widehat{v}^{(q)}(\xi) \approx 0$ for ξ in a neighborhood of t. In this case, r_S will be dominated by Δ. In this case, $r_S(h; v, t, \epsilon_f) < r_l$ for all h and h will thus monotonically increase until the maximum number of iterations is reached (which we set `max_iter` to 20). This occurs, for example, with any $(q-1)$-th degree polynomial. In this case, the method provides a warning but does not flag this as a failure. Note that in this case, h is a good choice because sending $h^* \to \infty$ would allow for indefinite reduction in the noise.

V. Extension to Standard Deviation. In some settings, $\epsilon(x)$ is modeled as a random variable, and we only have access to the standard deviation of the noise. Assuming $\mathbb{E}[\epsilon(x)] = 0$, one can extend this procedure to the stochastic setting by replacing ϵ_f with

$\sigma_f = \sqrt{\mathbb{E}[\epsilon(x)^2]}$. While the finite termination guarantees will not hold, particularly if the noise is unbounded, the procedure will yield an h that has the same dependence on σ_f and $\hat{v}^{(2)}(t)$ as the optimal finite-difference interval.

4.2.2.2. Finite Termination. Next, we prove a finite termination theorem for Algorithm 4.1. We start by making the following assumptions:

Assumption 4.2.1. *The testing ratio r_S satisfies:*

$$|r_S(h;\hat{v},t,\epsilon_f) - r_S(h;v,t,\epsilon_f)| \leq 1,\ \forall x \in \mathbb{R}, \forall h > 0$$

This assumption is satisfied by our requirement that $\|\tilde{w}\|_1 = 1$ and that $|\epsilon(t)| \leq \epsilon_f$.

Assumption 4.2.2. *As a function of h, $r_S(h;\hat{v},t,\epsilon_f)$ is continuous with $r_S(0;\hat{v},t,\epsilon_f) = 0$, and there exists an integer $K \in \mathbb{N}$ such that*

$$r_S(2^K h_0;\hat{v},t,\epsilon_f) \geq r_u - 1$$

Assuming that $\epsilon_f > 0$, the requirement that $r_S(0;\hat{v},t,\epsilon_f) = 0$ is satisfied because we have (4.14). While this condition is technical, it can be intuitively understood as an assumption to rule out the special case mentioned above where $\hat{v}^{(q)}(\xi) \approx 0$ for ξ in a neighborhood of t, and it is satisfied if, for example, we have $|\hat{v}^{(q)}(\xi)| \geq \eta > 0$ for all $\xi \in \mathbb{R}$.

Theorem 4.2.1. *Suppose Assumptions 4.2.1 and 4.2.2 are satisfied. In addition, suppose r_u and r_l are chosen such that $0 < r_l < r_u - 2$. Then, Algorithm 4.1 will terminate successfully in a finite number of iterations.*

Proof. Throughout this proof, we assume $h \geq 0$.

We will prove by this contradiction. Suppose Algorithm 4.1 does not terminate finitely. We denote the variables l, u, h used *at the beginning of* the k-th iteration of Algorithm 4.1 as l_k, u_k, h_k, respectively. Obviously, we have

$$0 \leq l_k \leq h_k \leq u_k, \ \forall k \in \mathbb{N},$$

and

$$l_k \leq l_{k+1} < u_{k+1} \leq u_k, \ \forall k \in \mathbb{N}.$$

First, we show that $r_S(l_k; \hat{v}, t, \epsilon_f) < r_l + 1$ for all $k \in \mathbb{N}$, by induction on k. Clearly this is true for $k = 0$ since $l_0 = 0$, and we have $r_S(0; \hat{v}, t, \epsilon_f) = 0$ by Assumption 4.2.2. Suppose this is true for $k \leq K$. We have two cases: (1) $r_S(h_K; v, t, \epsilon_f) < r_l$, which by Assumption 4.2.1 implies $r_S(h_K; \hat{v}, t, \epsilon_f) \leq r_S(h_K; v, t, \epsilon_f) + 1 < r_l + 1$. In this case $l_{K+1} = h_K$, so $r_S(l_{K+1}; \hat{v}, t, \epsilon_f) = r_S(h_K; \hat{v}, t, \epsilon_f) < r_l + 1$. (2) $r_S(h_K; v, t, \epsilon_f) > r_u$, in which case $l_{K+1} = l_K$ so by the induction hypothesis $r_S(l_{K+1}; \hat{v}, t, \epsilon_f) = r_S(l_K; \hat{v}, t, \epsilon_f) < r_l + 1$. Therefore the induction hypothesis holds for $K + 1$-th iteration.

By a similar argument, we can show that either $u_k = +\infty$, or $u_k < +\infty$ and $r_S(u_k; \hat{v}, t, \epsilon_f) > r_u - 1$ for all $k \in \mathbb{N}$.

In summary, we can show that for all $k \in \mathbb{N}$, we have

$$\text{either } r_S(l_k; \hat{v}, t, \epsilon_f) < r_l + 1 < r_u - 1 < r_S(u_k; \hat{v}, t, \epsilon_f), \tag{4.27}$$

$$\text{or } r_S(l_k; \hat{v}, t, \epsilon_f) < r_l + 1 \text{ and } u_k = +\infty. \tag{4.28}$$

Next, we claim that there exists $K_1 \in \mathbb{N}$ such that $u_k < +\infty$ for $k \geq K_1$. Suppose this is not the case, then we have $r_S(h_k; v, t, \epsilon_f) < r_l,\ \forall k \in \mathbb{N}$. In this case, we have $h_{k+1} = 2l_{k+1} = 2h_k$, so $h_k = 2^k h_0$ for all $k \in \mathbb{N}$. By Assumption 4.2.2, there exists $K \in \mathbb{N}$ such that $r_S(h_K; \hat{v}, t, \epsilon_f) \geq r_u - 1$, and since $r_S(h_K; v, t, \epsilon_f) \geq r_S(h_K; \hat{v}, t, \epsilon_f) - 1$, we have $r_S(h_K; v, t, \epsilon_f) \geq r_u - 2 > r_l$, a contradiction to $r_S(h_k; v, t, \epsilon_f) < r_l,\ \forall k \in \mathbb{N}$. This proves the existence of K_1.

Now we're ready to present the contradiction. For $k \geq K_1$, since $u_k < \infty$, we have

$$u_{k+1} - l_{k+1} = \frac{1}{2}(u_k - l_k)$$

This implies that $u_k - l_k \to 0$. Since $r_S(h; \hat{v}, t, \epsilon_f)$ (as a function of h) is continuous and $u_{K_1} < +\infty$, $[0, u_{K_1}]$ is compact so $r_S(h; \hat{v}, t, \epsilon_f)$ (as a function of h) is *uniformly continuous* on $[0, u_{K_1}]$. Note that $l_k, u_k \in [0, u_{K_1}]$ for $k \geq K_1$, therefore we have

$$r_S(u_k; \hat{v}, t, \epsilon_f) - r_S(l_k; \hat{v}, t, \epsilon_f) \to 0$$

This contradicts the fact that

$$r_S(l_k; \hat{v}, t, \epsilon_f) < r_l + 1 < r_u - 1 < r_S(u_k; \hat{v}, t, \epsilon_f),\ \forall k \in \mathbb{N},\ k \geq K_1$$

Therefore, Algorithm 4.1 must terminate finitely. This finishes the proof. □

4.2.3. Optimality of h

We finish this section by providing a guarantee for the optimality of h output by Algorithm 4.1 under certain assumptions.

Assumption 4.2.3. *Function $\hat{v} : \mathbb{R} \to \mathbb{R}$ is q-th continuously differentiable, with*

$$0 < L_q^l \leq \left|\hat{v}^{(q)}(\xi)\right| \leq L_q^u, \ \forall \xi \in \mathbb{R}$$

Notice that under Assumption 4.2.3, $\hat{v}^{(q)}(\xi)$ is a continuous function, and therefore it is implied that either $\hat{v}^{(q)}(\xi) < 0$ or $\hat{v}^{(q)}(\xi) > 0$ holds for all $\xi \in \mathbb{R}$. While this assumption may seem too strong to be useful, we actually only need such a bound to hold in a neighborhood of t. Here we make this stronger assumption to avoid complications and simplify the proof.

Theorem 4.2.2. *Suppose $\hat{v}$ satisfies Assumption 4.2.3. Consider estimating $\hat{v}^{(d)}(t)$ using a finite-difference scheme $S = (w, s)$. Then, the error in the estimator satisfies*

$$\left|v_S^{(d)}(t; h) - \hat{v}^{(d)}(t)\right| \leq |c_q| \, L_q^u h^{q-d} + \|w\|_1 \epsilon_f h^{-d},$$

and this bound is tight. The optimal h that minimizes this bound is

$$h^* = \left| \frac{d}{q-d} \frac{\|w\|_1 \epsilon_f}{c_q L_q^u} \right|^{1/q} \tag{4.29}$$

and the optimal worst-case error is of order

$$O\left(\epsilon_f^{\frac{q-d}{q}} \left(L_q^u\right)^{\frac{d}{q}} \right)$$

Proof. Let

$$\Gamma(h) = \sum_{j=1}^{p} w_j \cdot \hat{v}(t + h s_j)$$

Since $\hat{v}$ satisfies Assumption 4.2.3, $\Gamma(h)$ is q-th continuously differentiable. We apply Taylor's theorem to $\Gamma(h)$ at $h = 0$ with order $q - 1$. We have:

$$\Gamma(h) = \hat{v}^{(d)}(t)h^d + \frac{1}{q!}\Gamma^{(q)}(\theta)h^q, \text{ for some } \theta \in [0, h]$$

Note that

$$\Gamma^{(q)}(\theta) = \sum_{j=1}^{p} w_j s_j^q \cdot \hat{v}^{(q)}(t + hs_j).$$

By Assumption 4.2.3, we have

$$\left|\Gamma(h) - \hat{v}^{(d)}(t)h^d\right| \leq |c_q| \, L_q^u h^q$$

Now, notice that

$$v_S^{(d)}(t; h) = \frac{\Gamma(h) + \sum_{j=1}^{p} w_j \epsilon(t + hs_j)}{h^d},$$

therefore we have

$$\begin{aligned}
\left|v_S^{(d)}(t; h) - \hat{v}^{(d)}(t)\right| &= \left|\frac{\Gamma(h) + \sum_{j=1}^{p} w_j \epsilon(t + hs_j)}{h^d} - \hat{v}^{(d)}(t)\right| \\
&\leq \left|\Gamma(h) - \hat{v}^{(d)}(t)h^d\right| h^{-d} + \left|\sum_{j=1}^{p} w_j \epsilon(t + hs_j)\right| h^{-d} \\
&\leq |c_q| \, L_q^u h^{q-d} + \|w\|_1 \epsilon_f h^{-d}
\end{aligned}$$

To see the bound is tight, simply consider $\hat{v}(t) = \frac{1}{q!} L_q^u t^q$. The rest of the statement is obvious. □

Theorem 4.2.3. *Suppose $\hat{v}$ satisfies Assumption 4.2.3.Let $h_\dagger$ be the output of Algorithm 4.1. Then it holds that:*

$$h_\dagger \in \left[\left(\frac{r_l - 1}{|c_t|} \frac{\epsilon_f}{L_q^u} \right)^{1/q}, \left(\frac{r_u + 1}{|c_t|} \frac{\epsilon_f}{L_q^l} \right)^{1/q} \right] \tag{4.30}$$

Proof. Let

$$\Omega(h) = \sum_{j=1}^{\tilde{p}} \tilde{w}_j \cdot \hat{v}(t + h\tilde{s}_j)$$

We can see that $\Omega(h)$ is q-th continuously differentiable. We apply Taylor's theorem to $\Omega(h)$ at 0 with order $q-1$. By (4.14), the Taylor expansion of $\Omega(h)$ up to order $q-1$ is 0, so we're only left with the remainder term, i.e.,

$$\Omega(h) = \frac{1}{q!}\Omega^{(q)}(\theta)h^q, \text{ for some } \theta \in [0, h].$$

Note that

$$\Omega^{(q)}(\theta) = \sum_{j=1}^{\tilde{p}} \tilde{w}_j \tilde{s}_j^q \cdot \hat{v}^{(q)}(t + \theta\tilde{s}_j).$$

As we mentioned above, Assumption 4.2.3 implies that $\hat{v}^{(q)}(\xi) < 0$ or $\hat{v}^{(q)}(\xi) > 0$ for all $\xi \in \mathbb{R}$. Therefore, we have

$$|\Omega(h_\dagger)| \in \left[|c_t| \, L_q^l h_\dagger^q, \ |c_t| \, L_q^u h_\dagger^q \right].$$

Now, notice that

$$r_S(h) = \left| \frac{\Omega(h)}{\epsilon_f} + \Delta \right|.$$

Since $r_S(h_\dagger) \in [r_l, r_u]$ and $|\Delta| \leq 1$, we have:

$$|\Omega(h_\dagger)| \in \left[(r_l - 1)\epsilon_f, (r_u + 1)\epsilon_f \right].$$

This gives us:

$$(r_l - 1)\epsilon_f \le |c_t|\, L_q^u h_\dagger^q,\ (r_u + 1)\epsilon_f \ge |c_t|\, L_q^l h_\dagger^q,$$

i.e.,

$$h_\dagger \in \left[\left(\frac{r_l - 1}{|c_t|} \frac{\epsilon_f}{L_q^u} \right)^{1/q}, \left(\frac{r_u + 1}{|c_t|} \frac{\epsilon_f}{L_q^l} \right)^{1/q} \right]$$

□

Now, suppose both L_q^l and L_q^u are of the same order of magnitute $O(L_q)$, then we know from (4.29) that $h^* = O\left((\epsilon_f/L_q)^{1/q}\right)$, and from (4.30) that $h_\dagger = O\left((\epsilon_f/L_q)^{1/q}\right)$ as well, demonstrating the output of Algorithm 4.1 is optimal up to certain constant that depends on the scheme S and the testing ratio r_S.

4.3. Numerical Experiments

We test our proposed procedure on several univariate function. We focus on the case $d = 1$, i.e., gradient estimation, since this is most relevant to optimization. Throughout this section, we will test Algorithm 4.1 using 5 different estimating schemes, shown in Table 4.2. The testing ratio is generated using the *doubling trick*, i.e., using (4.25) with $\alpha = 2$.

label	s	w	q	Comment
`FD`	$(0, 1)$	$(-1, 1)$	2	forward difference
`CD`	$(-1, 1)$	$(-1/2, 1/2)$	3	central difference
`FD_3P`	$(0, 1, 2)$	$(-3/2, 2, -1/2)$	3	forward difference w/ 3 points
`FD_4P`	$(0, 1, 2, 3)$	$(-11/6, 3, -3/2, 1/1)$	4	forward difference w/ 4 points
`CD_4P`	$(-2, -1, 1, 2)$	$(1/12, -2/3, 2/3, -1/12)$	5	central difference w/ 4 points

Table 4.2. Schemes used in the experiments. The scheme is defined by $S = (w, s)$ as in (4.11). All schemes have $d = 1$ (i.e., estimating gradient); q is defined in (4.12).

For a specific testing function $\hat{v}$ at point t with noise ϵ_f and estimating scheme $S = (w, s)$, we plot the *worst case relative error*, as a function of differencing interval h, defined as:

$$\delta_S(h; \hat{v}, t, \epsilon_f) = \frac{1}{|\hat{v}^{(d)}(t)|} \left[\left| \frac{\sum_{j=1}^{p} w_j \hat{v}(t + s_j h)}{h^d} - \hat{v}^{(d)}(t) \right| + \|w\|_1 \frac{\epsilon_f}{h^d} \right]$$

This function captures the relative error of the estimation scheme S on function v at t *in the worst case*. The differencing interval h that minimizes $\delta_S(h; \hat{v}, t, \epsilon_f)$ is the optimal h. Notice that $\delta_S(h; \hat{v}, t, \epsilon_f)$ is a *deterministic* function that does not rely on the realization of actual noise in $v(xi)$.

We manually inject uniformly distributed, stochastic noise into $\hat{v}$ to obtain v, i.e., we have

$$v(\xi) = \hat{v}(\xi) + \epsilon(\xi), \ \epsilon(\xi) \sim \text{Uniform}(-\epsilon_f, \epsilon_f), \ \text{independent of everything else.}$$

We then perform our adaptive procedure Algorithm 4.1 on v and obtain an output $h_\dagger$. We will plot $h_\dagger$ and see how far it is from the minimizer of the worst case relative error function $\delta_S(h; \hat{v}, t, \epsilon_f)$. In the tables, we also report the optimizer that minimizes the function $\delta_S(h; \hat{v}, t, \epsilon_f)$; this value is returned by `scipy.optimize.minimize_scalar` function and might be unreliable.

4.3.1. Adaptivity to Noise Levels

To demonstrate our method is adaptive to different noise levels, we perform the test on a simple function $\hat{v}(t) = \cos(t)$ on different noise levels, using different schemes in Table 4.2. The results are shown in Figure 4.1, which demonstrate that our method can

automatically adjust to different noise levels and find near-optimal h. Detailed results on number of iterations taken and number of function evaluations can be found in Table 4.3.

4.3.2. Affine Invariance

One advantage of our proposed method is that it outputs correct h when the function v undergoes through an affine transformation. By the way we define our testing ratio, it is obvious that our procedure is invariant when adding a constant to the function. Hence, we focus on the transformation of the type $\hat{v}(t) \to a \cdot \hat{v}(b \cdot t)$ for some $a, b \neq 0$.

Specifically, we test Algorithm 4.1 on function $\hat{v}(t) = a \cdot \sin(b \cdot t)$ at $t = 0$, for different a and b. We fixed the noise level to be $\epsilon_f =$ `1E-3`. The results are shown in Figure 4.2. Detailed results can be found in Table 4.4. We can see that our method is affine-invariant and can output correct results for functions undergone affine transformations.

4.3.3. Difficult and Special Examples

In this subsection, we consider certain functions that are considered difficult or special in some sense, to demonstrate some features of our method. Some examples come from [**28**]. These examples include:

(1) $\hat{v}(t) = (e^t - 1)^2$, at $t = -8$. This function has extremely small first and second-order derivative at $t = -8$, but quickly increases as t increases beyond $t = 0$; a naive choice of $h = \sqrt{\epsilon_f / L_2}$ for forward difference can result in an extremely large h and lead to huge error.

(2) $\hat{v}(t) = e^{100t}$, at $t = 0.01$.

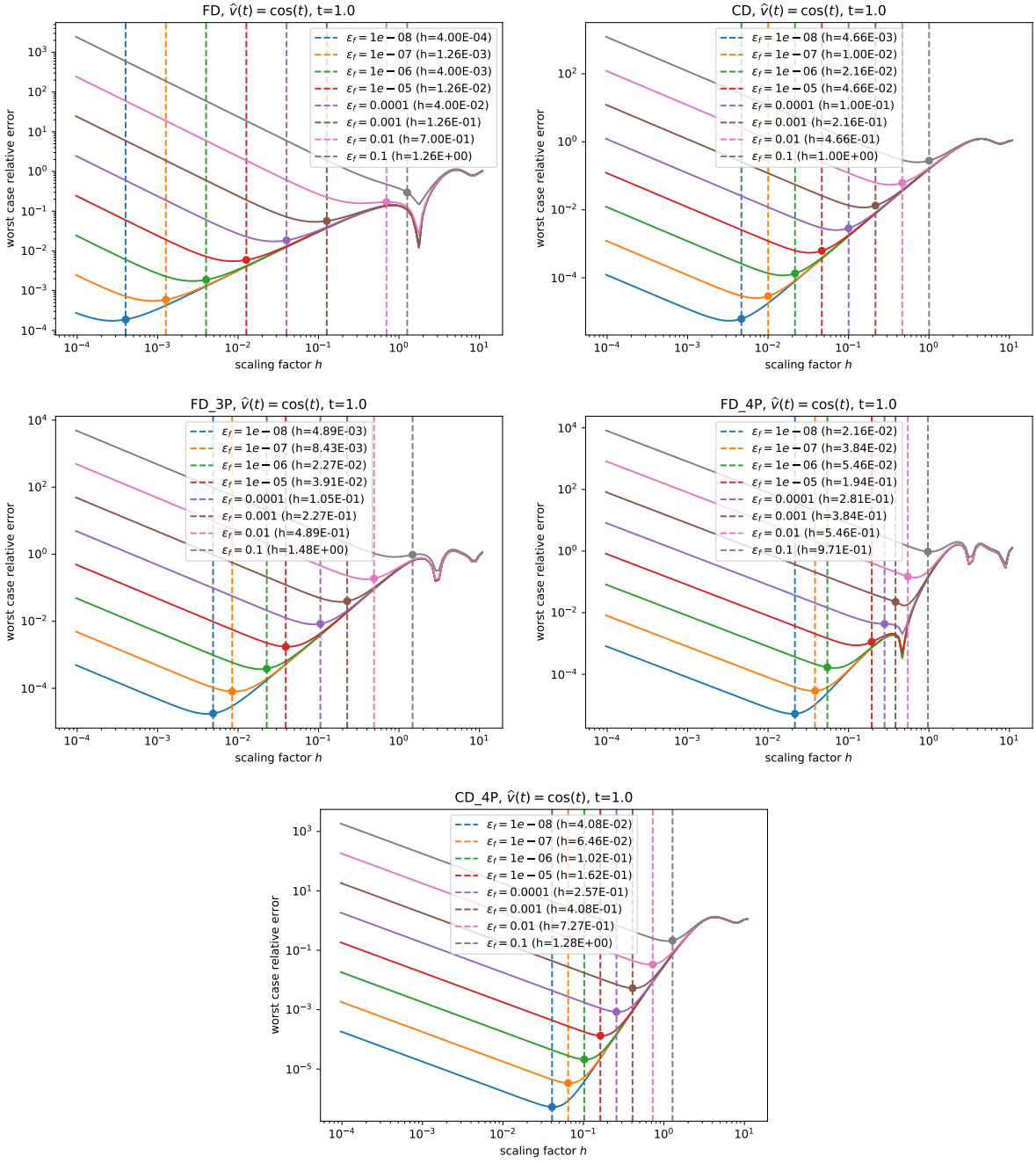

Figure 4.1. Worst case relative error $\delta_S(h; \hat{v}, t, \epsilon_f)$ against h on function $\hat{v}(t) = \cos(t)$ with different noise levels; the vertical dashed line represents the $h_\dagger$ output by Algorithm 4.1.

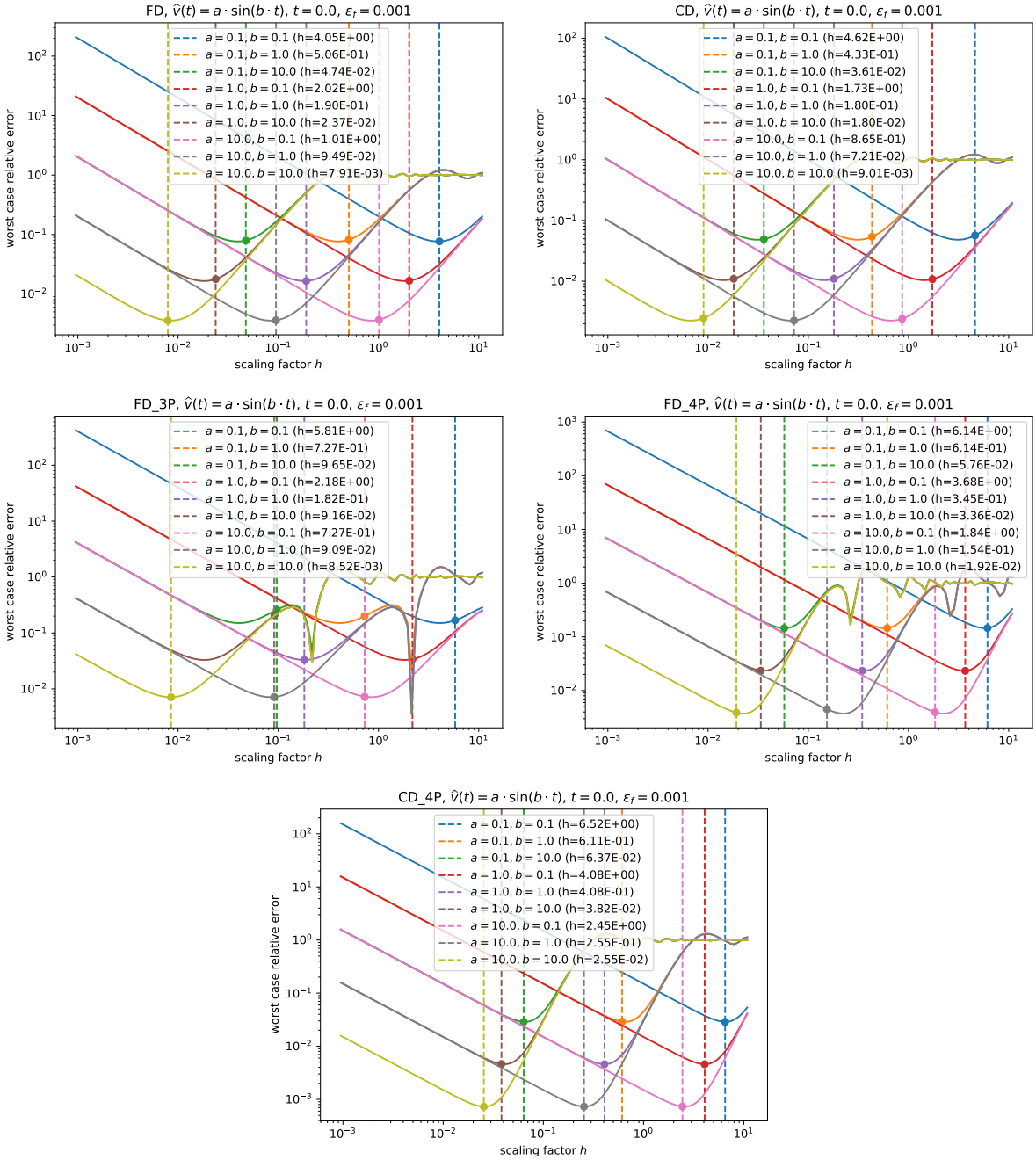

Figure 4.2. Worst case relative error $\delta_S(h; \hat{v}, t, \epsilon_f)$ against h on function $\hat{v}(t) = a \cdot \sin(b \cdot t)$ for different a and b; the vertical dashed line represents the $h_\dagger$ output by Algorithm 4.1.

scheme	$h_\dagger$	h^*	r	`n_iter`	`num_eval`	relative error	ϵ_f
`FD`	4.00e-04	2.74e-04	2.02	2	4	1.46e-04	1.00e-08
`FD`	1.26e-03	8.60e-04	2.30	2	4	3.61e-04	1.00e-07
`FD`	4.00e-03	2.73e-03	1.93	2	4	1.34e-03	1.00e-06
`FD`	1.26e-02	8.64e-03	2.53	2	4	3.61e-03	1.00e-05
`FD`	4.00e-02	2.76e-02	2.34	2	4	1.16e-02	1.00e-04
`FD`	1.26e-01	1.73e+00	1.51	2	4	3.72e-02	1.00e-03
`FD`	7.00e-01	8.24e+00	1.66	5	9	1.32e-01	1.00e-02
`FD`	1.26e+00	8.26e+00	2.09	2	4	1.19e-01	1.00e-01
`CD`	4.66e-03	3.29e-03	2.58	3	10	4.26e-06	1.00e-08
`CD`	1.00e-02	7.09e-03	2.65	3	10	2.13e-05	1.00e-07
`CD`	2.16e-02	1.53e-02	3.05	3	10	5.71e-05	1.00e-06
`CD`	4.66e-02	3.29e-02	3.06	3	10	2.05e-04	1.00e-05
`CD`	1.00e-01	7.09e-02	2.40	3	10	2.24e-03	1.00e-04
`CD`	2.16e-01	1.53e-01	2.80	3	10	8.22e-03	1.00e-03
`CD`	4.66e-01	7.73e+00	2.84	3	10	2.62e-02	1.00e-02
`CD`	1.00e+00	7.74e+00	2.57	3	10	7.78e-02	1.00e-01
`FD_3P`	4.89e-03	4.14e-03	2.04	4	11	9.66e-06	1.00e-08
`FD_3P`	8.43e-03	8.92e-03	1.26	1	4	2.01e-05	1.00e-07
`FD_3P`	2.27e-02	1.92e-02	2.75	4	11	8.18e-05	1.00e-06
`FD_3P`	3.91e-02	4.11e-02	1.31	1	4	5.82e-04	1.00e-05
`FD_3P`	1.05e-01	8.77e-02	3.07	4	11	1.66e-03	1.00e-04
`FD_3P`	2.27e-01	2.99e+00	1.97	4	11	2.62e-02	1.00e-03
`FD_3P`	4.89e-01	2.99e+00	2.20	4	11	8.34e-02	1.00e-02
`FD_3P`	1.48e+00	2.12e+01	1.83	4	11	7.25e-01	1.00e-01
`FD_4P`	2.16e-02	2.04e-02	2.08	4	16	2.14e-06	1.00e-08
`FD_4P`	3.84e-02	4.76e-01	2.17	4	16	7.35e-06	1.00e-07
`FD_4P`	5.46e-02	6.68e-02	1.13	1	6	3.53e-05	1.00e-06
`FD_4P`	1.94e-01	4.76e-01	2.42	2	8	6.70e-04	1.00e-05
`FD_4P`	2.81e-01	3.28e+00	1.79	5	21	1.28e-03	1.00e-04
`FD_4P`	3.84e-01	3.28e+00	1.57	4	16	4.12e-03	1.00e-03
`FD_4P`	5.46e-01	8.78e+00	1.31	1	6	3.19e-02	1.00e-02
`FD_4P`	9.71e-01	8.79e+00	1.41	1	6	6.66e-02	1.00e-01
`CD_4P`	4.08e-02	4.22e-02	2.52	1	6	1.16e-07	1.00e-08
`CD_4P`	6.46e-02	6.69e-02	2.04	1	6	8.34e-07	1.00e-07
`CD_4P`	1.02e-01	1.06e-01	1.95	1	6	8.81e-06	1.00e-06
`CD_4P`	1.62e-01	1.68e-01	1.79	1	6	4.29e-05	1.00e-05
`CD_4P`	2.57e-01	2.67e-01	1.71	1	6	4.00e-04	1.00e-04
`CD_4P`	4.08e-01	4.25e-01	1.89	1	6	8.32e-04	1.00e-03
`CD_4P`	7.27e-01	7.97e+00	2.87	5	26	7.57e-03	1.00e-02
`CD_4P`	1.28e+00	2.06e+01	2.03	4	20	1.92e-01	1.00e-01

Table 4.3. Detailed results for $\hat{v}(t) = \cos(t)$ with different noise levels; r represents the final testing ratio; h^* is the h that minimizes $\delta_S(h; \hat{v}, t, \epsilon_f)$ reported by `minimize_scalar` function in `scipy.optimize` and could be unreliable.

(3) $\hat{v}(t) = t^4 + 3t^2 - 10t$, at $t = 0.99999$. This function is considered difficult because $\hat{v}'(1) = 0$, and represents a case where the estimated derivative is very close to 0. In addition, this function is a 4-th order polynomial, so the optimal h for `CD_4P` is $+\infty$.

(4) $\hat{v}(t) = 10000t^3 + 0.01t^2 + 5t$, at $t =$ `1E-9`. This example is difficult in that it has approximate central symmetry at $t = 0$, which can lead to issues for adaptive procedures like proposed in [**28**]. However, we will show that this will not cause problems for our method.

For each example, we again fix $\epsilon_f =$ `1E-3`, and perform our estimation procedure for different schemes, and plot the worst case relative error. The results can be found in Figure 4.3 and Table 4.5.

It is intereting to observe the results for the two polynomials. For $\hat{v}(t) = t^4 + 3t^2 - 10t$ and scheme `CD_4P`, our procedure generated a huge $h_\dagger$; this is consistent with the fact that scheme `CD_4P` has $q = 5$, and $\hat{v}^{(5)}(\xi) = 0$ for all ξ on this example, which implies that we should choose h to be as large as possible. The similar is true for scheme `FD_4P` and `CD_4P` on function $\hat{v}(t) = 10000t^3 + 0.01t^2 + 5t$.

More interestingly, while theoretically speaking we should choose $h = \infty$ in such cases, we can see in Figure 4.3 that this is not the case. There exists a large h such that $\delta_S(h; \hat{v}, t, \epsilon_f)$ is minimized, beyond which the relative error begins to increase, and our method can successfully identify this h. The reason for this phenomenon is round-off errors; when h becomes too large, round-off error which is multiplicative will dominate ϵ_f; approximately, this will happen when $\max_j |\hat{v}(t + s_j h)| \, \epsilon_M$ becomes comparable to ϵ_f, and this is where the optimal h lies.

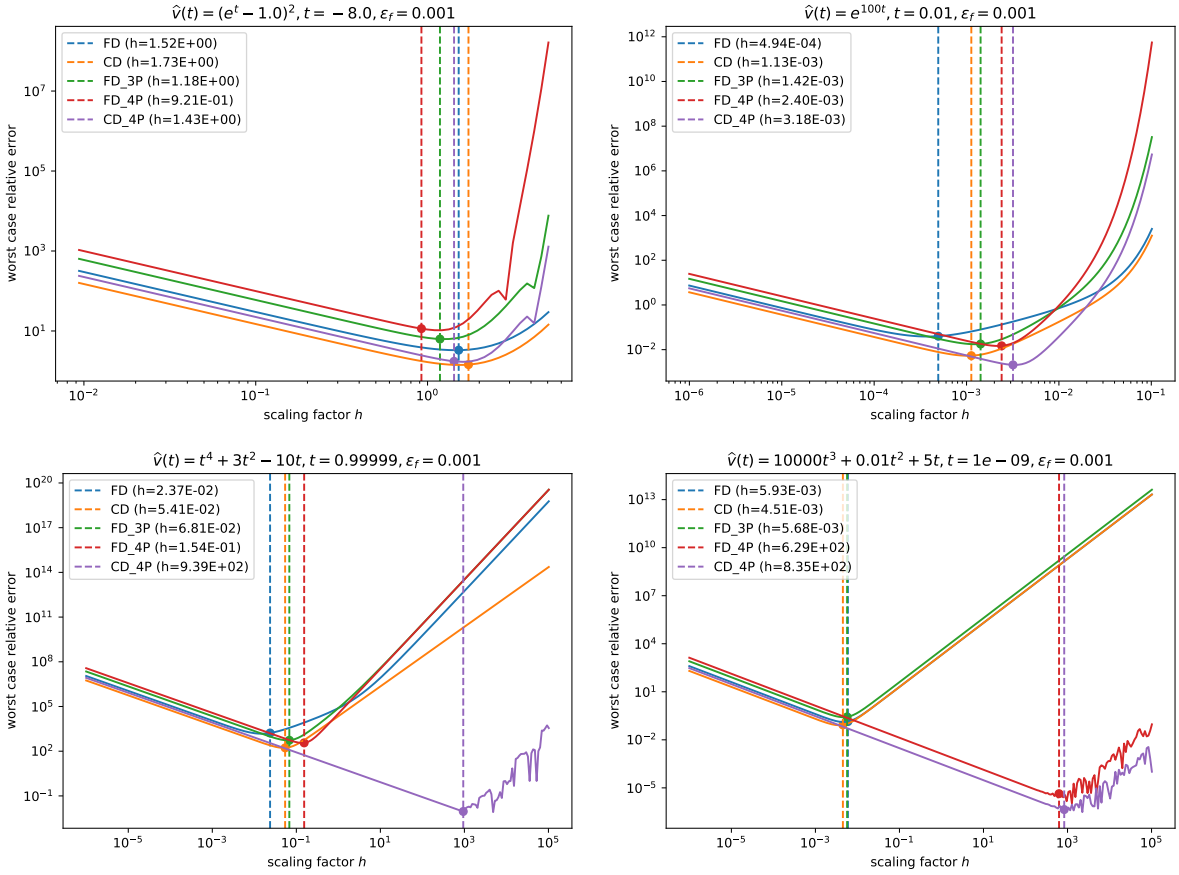

Figure 4.3. Worst case relative error $\delta_S(h; \hat{v}, t, \epsilon_f)$ against h on several special cases; the vertical dashed line represents the $h_\dagger$ output by Algorithm 4.1.

4.4. Conclusion

In this work, we propose a general, adaptive procedure for estimating differencing interval for a general finite difference estimation scheme. Without the need for any information about higher-order derivatives of the function, the proposed procedure can automatically produce an differencing interval h that is optimal up to a constant. Our numerical experiments demonstrate that the proposed method is both reliable and efficient for a wild range of noise levels and functions. We believe such an adaptive procedure lays

the fundation for developing efficient and robust finite-difference based DFO optimization methods.

a	b	scheme	$h_\dagger$	h^*	r	n_iter	num_eval	relative error
0.10	0.10	FD	4.05e+00	3.94e+00	1.45	7	9	3.59e-02
0.10	1.00	FD	5.06e-01	7.73e+00	3.20	4	6	3.60e-02
0.10	10.00	FD	4.74e-02	7.73e-01	2.39	3	6	4.69e-02
1.00	0.10	FD	2.02e+00	1.82e+00	2.01	6	8	8.11e-03
1.00	1.00	FD	1.90e-01	1.82e-01	1.38	4	7	1.02e-02
1.00	10.00	FD	2.37e-02	1.82e-02	2.76	4	7	1.48e-02
10.00	0.10	FD	1.01e+00	8.44e-01	2.63	5	7	1.75e-03
10.00	1.00	FD	9.49e-02	8.44e-02	2.13	3	6	1.93e-03
10.00	10.00	FD	7.91e-03	8.44e-03	1.53	4	6	5.66e-04
0.10	0.10	CD	4.62e+00	7.73e+01	3.11	6	14	3.51e-02
0.10	1.00	CD	4.33e-01	3.12e-01	2.58	4	12	3.09e-02
0.10	10.00	CD	3.61e-02	3.12e-02	1.51	3	8	2.15e-02
1.00	0.10	CD	1.73e+00	1.44e+00	1.72	6	16	4.98e-03
1.00	1.00	CD	1.80e-01	1.44e-01	1.94	4	14	5.41e-03
1.00	10.00	CD	1.80e-02	1.44e-02	1.94	4	10	5.41e-03
10.00	0.10	CD	8.65e-01	6.70e-01	2.16	5	14	1.25e-03
10.00	1.00	CD	7.21e-02	6.70e-02	1.25	2	6	8.66e-04
10.00	10.00	CD	9.01e-03	6.70e-03	2.44	5	12	1.35e-03
0.10	0.10	FD_3P	5.81e+00	8.14e+01	2.45	6	9	6.06e-02
0.10	1.00	FD_3P	7.27e-01	8.14e+00	2.91	3	6	1.12e-01
0.10	10.00	FD_3P	9.65e-02	8.14e-01	1.59	6	17	1.99e-01
1.00	0.10	FD_3P	2.18e+00	1.83e+00	1.59	6	11	2.22e-02
1.00	1.00	FD_3P	1.82e-01	2.14e+00	1.47	1	4	1.86e-04
1.00	10.00	FD_3P	9.16e-02	8.14e-01	2.67	9	26	2.05e-01
10.00	0.10	FD_3P	7.27e-01	8.45e-01	1.47	3	6	1.54e-03
10.00	1.00	FD_3P	9.09e-02	8.45e-02	1.62	2	5	1.77e-03
10.00	10.00	FD_3P	8.52e-03	8.45e-03	1.85	7	12	7.43e-04
0.10	0.10	FD_4P	6.14e+00	8.44e+01	2.91	8	23	4.81e-03
0.10	1.00	FD_4P	6.14e-01	8.44e+00	2.82	2	8	5.25e-03
0.10	10.00	FD_4P	5.76e-02	8.44e-01	2.48	5	15	1.79e-02
1.00	0.10	FD_4P	3.68e+00	2.71e+01	2.58	6	16	1.33e-02
1.00	1.00	FD_4P	3.45e-01	2.71e+00	2.15	5	20	2.42e-03
1.00	10.00	FD_4P	3.36e-02	2.71e-01	2.04	7	22	9.53e-04
10.00	0.10	FD_4P	1.84e+00	2.25e+00	1.45	5	14	3.68e-04
10.00	1.00	FD_4P	1.54e-01	2.25e-01	1.12	2	8	2.81e-03
10.00	10.00	FD_4P	1.92e-02	2.71e-01	1.36	5	14	7.83e-04
0.10	0.10	CD_4P	6.52e+00	7.97e+01	2.12	5	14	5.73e-03
0.10	1.00	CD_4P	6.11e-01	7.97e+00	1.57	3	14	4.45e-03
0.10	10.00	CD_4P	6.37e-02	7.97e-01	1.90	6	24	5.23e-03
1.00	0.10	CD_4P	4.08e+00	4.10e+00	2.30	7	26	9.02e-04
1.00	1.00	CD_4P	4.08e-01	4.10e-01	2.30	1	6	9.02e-04
1.00	10.00	CD_4P	3.82e-02	4.10e-02	1.68	6	20	6.98e-04
10.00	0.10	CD_4P	2.45e+00	2.58e+00	1.89	5	18	1.18e-04
10.00	1.00	CD_4P	2.55e-01	2.58e-01	2.31	4	20	1.39e-04
10.00	10.00	CD_4P	2.55e-02	2.58e-02	2.31	5	14	1.39e-04

Table 4.4. Detailed results for $\hat{v}(t) = a{\cdot}\sin(b{\cdot}t)$ with $\epsilon_f =$ 1E-3; r represents the final testing ratio; h^* is the h that minimizes $\delta_S(h; \hat{v}, t, \epsilon_f)$ reported by minimize_scalar function in scipy.optimize and could be unreliable.

$\hat{v}(t)$	scheme	$h_\dagger$	h^*	r	n_iter	num_eval	relative error
$(e^t - 1.0)^2$	FD	1.52e+00	1.46e+00	2.30	7	10	6.32e-01
$(e^t - 1.0)^2$	CD	1.73e+00	8.69e+00	2.97	6	16	4.01e-02
$(e^t - 1.0)^2$	FD_3P	1.18e+00	3.82e+02	2.92	7	16	5.34e-01
$(e^t - 1.0)^2$	FD_4P	9.21e-01	3.82e+02	1.65	4	12	4.15e+00
$(e^t - 1.0)^2$	CD_4P	1.43e+00	3.82e+02	1.84	5	22	1.11e+00
e^{100t}	FD	4.94e-04	3.78e-04	2.31	8	10	1.84e-02
e^{100t}	CD	1.13e-03	1.03e-03	1.38	8	18	1.82e-03
e^{100t}	FD_3P	1.42e-03	3.82e+02	2.46	8	11	4.18e-03
e^{100t}	FD_4P	2.40e-03	3.82e+02	3.14	8	20	8.46e-03
e^{100t}	CD_4P	3.18e-03	3.82e+02	2.15	8	20	4.60e-04
$t^4 + 3t^2 - 10t$	FD	2.37e-02	1.48e-02	2.66	4	7	1.19e+03
$t^4 + 3t^2 - 10t$	CD	5.41e-02	5.00e-02	1.11	4	12	1.08e+02
$t^4 + 3t^2 - 10t$	FD_3P	6.81e-02	6.16e-02	1.97	4	9	2.67e+02
$t^4 + 3t^2 - 10t$	FD_4P	1.54e-01	1.39e-01	2.65	2	8	1.93e+02
$t^4 + 3t^2 - 10t$	CD_4P	9.39e+02	4.87e+03	1.62	16	48	2.71e-03
$10000t^3 + 0.01t^2 + 5t$	FD	5.93e-03	4.64e-03	2.79	6	9	1.00e-01
$10000t^3 + 0.01t^2 + 5t$	CD	4.51e-03	3.68e-03	2.19	6	14	2.65e-02
$10000t^3 + 0.01t^2 + 5t$	FD_3P	5.68e-03	4.64e-03	2.73	6	9	8.46e-02
$10000t^3 + 0.01t^2 + 5t$	FD_4P	6.29e+02	4.13e+03	1.95	12	28	1.59e-06
$10000t^3 + 0.01t^2 + 5t$	CD_4P	8.35e+02	1.03e+04	1.95	12	28	1.99e-07

Table 4.5. Detailed results for special examples, with ϵ_f = 1E-3; r represents the final testing ratio; h^* is the h that minimizes $\delta_S(h; \hat{v}, t, \epsilon_f)$ reported by minimize_scalar function in scipy.optimize and could be unreliable.

CHAPTER 5

Constrained and Composite Optimization via Adaptive Sampling Methods

5.1. Introduction

In this chapter, we study the solution of constrained and composite optimization problems in which the objective function is stochastic and the constraints or regularizers are deterministic. We propose methods that automatically adjust the quality of the gradient estimate so as to keep computational cost at a minimum while ensuring a fast rate of convergence. Methods of this kind have been studied in the context of unconstrained optimization but their extension to the constrained and composite optimization settings is not simple because the projections or proximal operators used in the methods introduce discontinuities. This renders existing rules for the control of the gradient unreliable. Whereas in the unconstrained setting pointwise decisions suffice to estimate the quality of a gradient approximation, in the presence of constraints or nonsmooth regularizers one must analyze the result of a complete step.

Let us begin by considering the optimization problem

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{s.t.} \quad x \in \Omega, \tag{5.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a stochastic objective function and Ω is a deterministic convex closed set. Automatic rules for controlling the quality of the gradient when $\Omega = \mathbb{R}^n$ have been

studied from a theoretical perspective and have been successfully applied to expected risk minimization problems arising in machine learning. Since in that context the gradient approximation is controlled by the sample size, these methods have been called "adaptive sampling" methods. A fundamental mechanism for controlling the quality of the gradient in the unconstrained setting is the *norm test* [**16**], which lies behind most algorithms and theory of adaptive sampling methods.

To describe this test, let $\Omega = \mathbb{R}^n$, and consider the iteration

$$x_{k+1} = x_k - \alpha_k g_k, \tag{5.2}$$

where $\alpha_k > 0$ is a steplength and g_k is an approximation to the gradient $\nabla f(x_k)$. To determine if g_k is sufficiently accurate to ensure that iteration (5.2) is convergent, one can test the inequality [**16**]:

$$\mathbb{E}[\|g_k - \nabla f(x_k)\|_2^2] \leq \xi \|\nabla f(x_k)\|_2^2, \quad \xi > 0, \tag{5.3}$$

where the expectation is taken with respect to the choice of g_k at iteration k. If (5.3) is satisfied, g_k is deemed accurate enough; otherwise a new and more accurate gradient approximation is computed. We refer to this procedure as the norm test to distinguish it from tests based on angles [**14**].

The norm test is, however, not adequate in the constrained setting. To see this, suppose that we apply the gradient projection method, $x_{k+1} = P_\Omega[x_k - \alpha_k g_k]$, to solve problem (5.1) when $\Omega \neq \mathbb{R}^n$. A condition such as (5.3) on the quality of the gradient approximation at one point cannot always predict the quality of the full step because the latter is based on a projection of the gradient, which may be much smaller. This

is illustrated in Figure 5.1, where we consider the minimization of a strongly convex quadratic function subject to a linear constraint:

$$\min_{x\in\mathbb{R}^n} \tfrac{1}{2}x^TQx + b^Tx + c \qquad \text{s.t. } a^Tx \le 0.$$

In Figure 5.1, $\widehat{x}^*$ denotes the unconstrained minimizer and x^* the solution of the constrained problem. We let the iterate x_k lie on the boundary of the constraint, very close to the solution x^*, and observe that $\|\nabla f(x_k)\|$ is large, and stays large as x_k approaches x^*. Thus, (5.3) does not force the error in g_k to zero as $x_k \to x^*$.

The instance of g_k shown in Figure 5.1 satisfies $\|g_k - \nabla f(x_k)\| < \|\nabla f(x_k)\|$, but results in a poor step. Clearly satisfaction of (5.3) allows for many such steps. Note, however, that $\|g_k - \nabla f(x_k)\|$ is greater than the norm of the projected gradient $P_\Omega[g_k]$, which is a more appropriate measure. Thus, since we are concerned about the error in the total

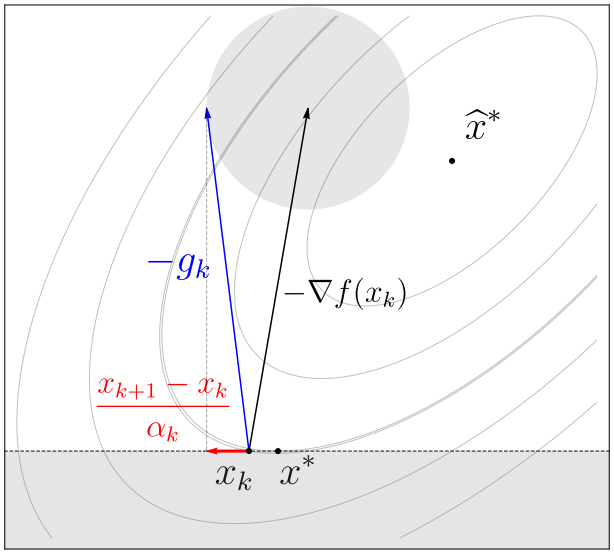

Figure 5.1. Failure of norm test for constrained problems.

step, and in this example the step is given by $x_{k+1} - x_k = -\alpha_k P_\Omega[g_k]$, it makes sense to compare $\|g_k - \nabla f(x_k)\|$ to $\|P_\Omega[g_k]\| = \|x_{k+1} - x_k\|/\alpha_k$.

We generalize this idea and propose the following procedure for measuring the quality of the gradient approximation. We first compute a projected step $\overline{x}_{k+1} = P_\Omega[x_k - \alpha_k g_k]$ based on the current gradient estimate g_k, and regard g_k to be acceptable if the following inequality holds:

$$\mathbb{E}[\|g_k - \nabla f(x_k)\|_2^2] \equiv \mathrm{Var}_k\left[g_k\right] \leq \xi \left\| \frac{\mathbb{E}_k\left[\overline{x}_{k+1}\right] - x_k}{\alpha_k} \right\|_2^2, \quad \xi > 0. \tag{5.4}$$

Otherwise, we compute a more accurate gradient estimate g_k, and recompute the step to obtain the new iterate x_{k+1}.

In this strategy one must therefore look ahead, suggesting that a convenient framework for the design and analysis of adaptive sampling methods for constrained optimization is the *proximal gradient method*. In addition to its versatility, the proximal gradient method allows us to expand the range of our investigation to include the composite optimization problem

$$\min_{x \in \mathbb{R}^n} \ \phi(x) = f(x) + h(x), \tag{5.5}$$

where f is a stochastic function and h is a convex (but not necessarily smooth or finite-valued) function. The constrained optimization problem (5.1) can be written in the form (5.5) by defining h to be the convex indicator function for the set Ω.

The goal of this chapter is to design an adaptive mechanism for gradually improving the gradient accuracy that can be regarded as an extension of the norm test (5.3) to problems (5.1) and (5.5). We argue in Section 5.3 that the condition (5.4), with ϕ replacing f, can be

used to build such a mechanism within a proximal gradient method. Although condition (5.4) appears to be impractical since it involves $\mathbb{E}_k[\overline{x}_{k+1}]$, we show how to approximate it in practice. The proposed algorithm reacts to information observed during the course of the iteration, as opposed to methods that dictate the increase in the gradient size *a priori*. Specifically, it has been established in [**35**] that for a stochastic proximal gradient method in which the sample size grows like a^k, with $a > 1$, convergence can be assured in the convex case. However, the behavior of the algorithm depends very strongly on the value of a, and there are no clear guidelines on how to choose it for a given problem.

5.1.1. Literature Review

A deterministic version of the norm test was used by Carter [**20**] in the design of a trust region method for unconstrained optimization that employs inexact gradients. Friedlander and Schmidt [**27**] propose increasing the sample size geometrically for the solution of the finite-sum problem, establish a linear convergence result, and report numerical tests with a quasi-Newton method. Byrd et al. [**16**] studied the expected risk minimization problem and provide a complexity result for the geometric growth condition. That paper also introduces the stochastic version of the norm test (5.3), and reports results with a Newton-like method. Bollapragada et al. [**13**] introduced a variant of the norm test, called the the inner product test, which is designed to improve the practical efficiency of the method at the price of weakening the theoretical convergence guarantees. (An adaption of this test to problems (5.1) and (5.5) is presented in Section 5.4.) Adaptive sampling methods have also been studied by Cartis and Scheinberg [**21**], who establish a global rate of convergence of unconstrained optimization methods that (implicitly) satisfy the norm

condition. Pasupathy et al. [**49**] study sampling rates in stochastic recursions. Roosta et al. [**53, 54**] analyze sub-sampled Newton methods with adaptive sampling, and De et al. [**25**] study automatic inference with adaptive sampling.

There is a large literature on proximal gradient methods for solving composite optimization problems; see e.g. [**10**] and the references therein. Some of these studies consider inexact gradients [**55**], but these studies do not propose an automatic procedure for improving the quality of the gradient. An exception is [**3**], which deals with a similar subject, but differ from this chapter in various ways in its treatment of the topic.

5.2. Outline of the Algorithm

Since the constrained optimization problem (5.1) is a special case of the composite problem (5.5), we focus on the latter and state the problem under consideration as

$$\min_{x\in\mathbb{R}^n} \ \phi(x) = f(x) + h(x), \quad \text{where } f(x) = \mathbb{E}_{\theta\sim\Theta}\left[F(x,\theta)\right]. \tag{5.6}$$

Here, $F(\cdot,\theta) : \mathbb{R}^n \to \mathbb{R}$ is a smooth function, θ is a random variable with support Θ, and $h : \mathbb{R}^n \to \mathbb{R}\cup\{\infty\}$ is a convex (and generally nonsmooth) function. A popular method for solving composite optimization problems is the proximal gradient method (see e.g. [**11**]), which in the context of problem (5.6) is given as

$$x_{k+1} \leftarrow \operatorname*{argmin}_{x\in\mathbb{R}^n} \ f(x_k) + g_k^T(x - x_k) + \frac{1}{2\alpha_k}\|x - x_k\|^2 + h(x), \quad \text{with} \ \ 0 < \alpha_k \leq \tfrac{1}{L}, \tag{5.7}$$

where g_k is an unbiased estimator of $\nabla f(x_k)$ and L is a Lipschitz constant defined below. Here and henceforth, $\|\cdot\|$ denotes the Euclidean norm. As is well known, we can also

write this iteration as

$$x_{k+1} = \text{prox}_{\alpha_k h}\left(x_k - \alpha_k g_k\right), \tag{5.8}$$

where

$$\text{prox}_{\alpha_k h}(z_k) = \underset{x \in \mathbb{R}^d}{\text{argmin}}\ h(x) + \frac{1}{2\alpha_k}\left\|x - z_k\right\|^2. \tag{5.9}$$

The proposed adaptive sampling proximal gradient algorithm proceeds in two stages. At a given iterate x_k, it first computes a gradient approximation (using the current sample size) as well as a proximal gradient step. Based on information gathered from this step, it computes a second proximal gradient step that determines the new iterate x_{k+1}. An outline of this method is given in Algorithm 5.1.

Algorithm 5.1: *Outline of Adaptive Sampling Algorithm for Solving Problem* (5.6)

Input: x_0, sample size $S \in \mathbb{N}^+$, and sequence $\{\alpha_k > 0\}$.

1: **for** $k = 0, 1, 2, ...,$ **do**

2: Draw S i.i.d. samples $\{\theta_0, \theta_1, \cdots, \theta_{S-1}\}$ from Θ, compute

$$\overline{g}_k = \frac{1}{S} \sum_{i=0}^{S-1} \nabla_x F(x, \theta_i), \tag{5.10}$$

and the proximal gradient step

$$\overline{x}_{k+1} = \text{prox}_{\alpha_k h} \left(x_k - \alpha_k \overline{g}_k \right). \tag{5.11}$$

3: Determine the new sample size $S_k \geq S$ (see the next section).

4: **if** $S_k > S$ **then**

5: re-sample S_k i.i.d. samples $\left\{\hat{\theta}_0, \hat{\theta}_1, \cdots, \hat{\theta}_{S_k-1}\right\}$ from Θ, and compute:

$$x_{k+1} = \text{prox}_{\alpha_k h} \left(x_k - \alpha_k g_k \right), \text{ where } g_k = \frac{1}{S_k} \sum_{i=0}^{S_k-1} \nabla_x F(x, \theta_i);$$

6: **else**

7: let $x_{k+1} = \overline{x}_{k+1}$.

8: **end if**

9: Set $S \leftarrow S_k$.

10: **end for**

As discussed in Section 5.3.5, when $S_k > S$, one can reuse the samples from Step 1, and in Step 3 only gather $(S_k - S)$ additional i.i.d. samples $\left\{\hat{\theta}_S, \hat{\theta}_{S+1}, \cdots, \hat{\theta}_{S_k-1}\right\}$ from Θ.

The unspecified parts of this algorithm are the steplength sequence $\{\alpha_k\}$ and the determination of a sample size S_k in Step 2. The analysis in the next section provides the elements for making those decisions. One requirement of the strategy used in Step 2-3 is

that, when h is not present, Algorithm 1 should reduce to the iteration (5.2)-(5.3), i.e., to an adaptive sampling gradient method using the norm test to control the sample size.

5.3. Derivation of the Algorithm

To motivate our approach for determining the sample size S_k, we begin by deriving a fundamental condition (see (5.17) below) that ensures that the steps are good enough to ensure convergence in expectation. The rest of the derivation of the algorithm consist of devising a procedure for approximating condition (5.17) in practice.

5.3.1. A Fundamental Inequality

We recall that a function $f : \mathbb{R}^n \to \mathbb{R}$ is μ-strongly convex (with $\mu > 0$) iff

(5.12)

$$f(\gamma x + (1-\gamma)y) \leq \gamma f(x) + (1-\gamma)f(y) - \frac{\mu}{2}\gamma(1-\gamma)\left\|y - x\right\|^2, \ \forall x, y \in \mathbb{R}^n, \ \forall \gamma \in [0,1].$$

We also have that if f is a strongly convex and differentiable function, then

$$f(x) \geq f(y) + \nabla f(y)^T(x-y) + \frac{\mu}{2}\|x-y\|^2, \ \forall x \in \mathbb{R}^n. \tag{5.13}$$

If a function is continuously differentiable, μ-strongly convex, and has a Lipschitz continuous gradient with Lipschitz constant L, we say that f is $[\mu, L]$–smooth. We make the following assumptions about problem (5.6).

Assumption 5.3.1. *$f : \mathbb{R}^n \to \mathbb{R}$ is a $[\mu, L]$-smooth function and $h : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is a closed convex and proper function.*

These assumptions imply that the objective function ϕ defined in (5.6) is strongly convex, and we denote its minimizer by x^* and the minimum objective value by ϕ^*. We consider the stochastic proximal gradient method (5.7) where g_k is an unbiased estimator of $\nabla f(x_k)$ adapted to the filtration $\mathbb{T}$ generated as $\mathbb{T}_k = \sigma\left(x_0, g_0, g_1, \cdots, g_{k-1}\right)$. In other words, we assume that

$$\mathbb{E}\left(g_k|\mathbb{T}_k\right) = \nabla f(x_k). \tag{5.14}$$

For simplicity, we denote conditional expectation as $\mathbb{E}_k\left[\cdot\right] = \mathbb{E}\left(\cdot|\mathbb{T}_k\right)$ and conditional variance as $\mathrm{Var}_k\left[\cdot\right] = \mathbb{E}\left[\|\cdot\|^2|\mathbb{T}_k\right] - \|\mathbb{E}\left(\cdot|\mathbb{T}_k\right)\|^2$. In what follows, we let $f_k, \nabla f_k$ denote $f(x_k), \nabla f(x_k)$, and similarly for other functions. We begin by establishing a technical lemma that provides the first stepping stone in our analysis.

Lemma 5.3.1. *Suppose that Assumptions 5.3.1 hold and that $\{x_k\}$ is generated by iteration* (5.7)*, where g_k satisfies* (5.14)*. Then,*

$$\begin{aligned}\mathbb{E}_k\left[\phi_{k+1} - \phi^*\right] \leq (1 - \mu\alpha_k)(\phi_k - \phi^*) + \mathbb{E}_k\left[(\nabla f_k - g_k)^T(x_{k+1} - x_k)\right] \\ - \left(\frac{1}{2\alpha_k} - \frac{L}{2}\right)\mathbb{E}_k\left[\|x_{k+1} - x_k\|^2\right].\end{aligned}$$

Proof. By Assumptions 5.3.1, we have that for any fixed $x_k \in \mathbb{R}^n$,

$$\begin{aligned}
\phi_{k+1} &\leq f_k + \nabla f_k^T(x_{k+1} - x_k) + \frac{L}{2}\|x_{k+1} - x_k\|^2 + h_{k+1} \\
&= f_k + g_k^T(x_{k+1} - x_k) + \frac{1}{2\alpha_k}\|x_{k+1} - x_k\|^2 + h_{k+1} + (\nabla f_k - g_k)^T(x_{k+1} - x_k) \\
&\quad - \left(\frac{1}{2\alpha_k} - \frac{L}{2}\right)\|x_{k+1} - x_k\|^2 \\
&\leq f_k + g_k^T(x - x_k) + \frac{1}{2\alpha_k}\|x - x_k\|^2 + h(x) + (\nabla f_k - g_k)^T(x_{k+1} - x_k) \\
&\quad - \left(\frac{1}{2\alpha_k} - \frac{L}{2}\right)\|x_{k+1} - x_k\|^2 \quad \text{(for any } x \in \mathbb{R}^n \text{ by definition (5.7) of } x_{k+1}) \\
&= f_k + \nabla f_k^T(x - x_k) + \frac{1}{2\alpha_k}\|x - x_k\|^2 + h(x) + (\nabla f_k - g_k)^T(x_{k+1} - x_k) \\
&\quad + (g_k - \nabla f_k)^T(x - x_k) - \left(\frac{1}{2\alpha_k} - \frac{L}{2}\right)\|x_{k+1} - x_k\|^2 \\
&\leq \phi(x) + \left(\frac{1}{2\alpha_k} - \frac{\mu}{2}\right)\|x - x_k\|^2 + (\nabla f_k - g_k)^T(x_{k+1} - x_k) \\
&\quad + (g_k - \nabla f_k)^T(x - x_k) - \left(\frac{1}{2\alpha_k} - \frac{L}{2}\right)\|x_{k+1} - x_k\|^2 \quad \text{(by (5.13))}.
\end{aligned}$$

This inequality holds for any $x \in \mathbb{R}^n$. Let us substitute

$$x \leftarrow \widetilde{x}_k = \beta x^* + (1 - \beta)x_k, \quad \text{with} \;\; \beta = \mu\alpha_k, \tag{5.15}$$

in the relation above. Recalling the definition (5.12) of strong convexity, we obtain

$$\begin{aligned}\phi_{k+1} &\leq \phi(\widetilde{x}_k) + \left(\frac{1}{2\alpha_k} - \frac{\mu}{2}\right)\|\widetilde{x}_k - x_k\|^2 + (\nabla f_k - g_k)^T(x_{k+1} - x_k) \\ &\quad + (g_k - \nabla f_k)^T(\widetilde{x}_k - x_k) - \left(\frac{1}{2\alpha_k} - \frac{L}{2}\right)\|x_{k+1} - x_k\|^2 \\ &\leq \beta\phi^* + (1-\beta)\phi_k \underbrace{-\frac{\mu}{2}\beta(1-\beta)\|x^* - x_k\|^2 + \left(\frac{1}{2\alpha_k} - \frac{\mu}{2}\right)\|\widetilde{x}_k - x_k\|^2}_{\text{term 1}} \\ &\quad + (\nabla f_k - g_k)^T(x_{k+1} - x_k) + (g_k - \nabla f_k)^T(\widetilde{x}_k - x_k) - \left(\frac{1}{2\alpha_k} - \frac{L}{2}\right)\|x_{k+1} - x_k\|^2 .\end{aligned}$$

Term 1 can be written as

$$\begin{aligned}&-\frac{\mu}{2}\beta(1-\beta)\|x^* - x_k\|^2 + \left(\frac{1}{2\alpha_k} - \frac{\mu}{2}\right)\beta^2\|x^* - x_k\|^2 \\ &= -\frac{\mu}{2}\beta(1-\beta)\|x^* - x_k\|^2 + \left(\frac{1-\mu\alpha_k}{2\alpha_k}\right)\beta^2\|x^* - x_k\|^2 \\ &= \beta(1-\beta)\|x^* - x_k\|^2\left(\frac{\beta}{2\alpha_k} - \frac{\mu}{2}\right) \\ &= 0 \qquad (5.16)\end{aligned}$$

since $\beta = \mu\alpha_k$. Therefore,

$$\begin{aligned}\phi_{k+1} \leq \beta\phi^* + (1-\beta)\phi_k + (\nabla f_k - g_k)^T(x_{k+1} - x_k) + (g_k - \nabla f_k)^T(\widetilde{x}_k - x_k) \\ - \left(\frac{1}{2\alpha_k} - \frac{L}{2}\right)\|x_{k+1} - x_k\|^2 .\end{aligned}$$

Taking conditional expectation, noting that $\widetilde{x}_k \in \mathbb{T}_k$, and recalling (5.14), we have

$$\mathbb{E}_k\left[\phi_{k+1}\right] \leq \beta\phi^* + (1-\beta)\phi_k + \mathbb{E}_k\left[(\nabla f_k - g_k)^T(x_{k+1}-x_k)\right] - \left(\frac{1}{2\alpha_k} - \frac{L}{2}\right)\mathbb{E}_k\left[\|x_{k+1}-x_k\|^2\right],$$

and by the definition of β we conclude that

$$\mathbb{E}_k\left[\phi_{k+1}-\phi^*\right] \leq (1-\mu\alpha_k)(\phi_k-\phi^*) + \mathbb{E}_k\left[(\nabla f_k - g_k)^T(x_{k+1}-x_k)\right] - \left(\frac{1}{2\alpha_k} - \frac{L}{2}\right)\mathbb{E}_k\left[\|x_{k+1}-x_k\|^2\right].$$

□

From this result, we can readily establish conditions under which the proximal gradient iteration, with a fixed steplength $\alpha_k = \alpha$, achieves Q-linear convergence of ϕ_k, in expectation.

Theorem 5.3.1. *Suppose that Assumptions 5.3.1 hold, that $\{x_k\}$ is generated by (5.7) with $\alpha_k = (1-\eta)/L$ for $\eta \in (0,1)$, and that g_k satisfies (5.14). If we have that for all k,*

$$\alpha_k\mathbb{E}_k\left[(\nabla f_k - g_k)^T(x_{k+1}-x_k)\right] \leq \frac{\eta}{2}\mathbb{E}_k\left[\|x_{k+1}-x_k\|^2\right] \tag{5.17}$$

then

$$\mathbb{E}_k\left[\phi_{k+1}-\phi^*\right] \leq \left[1-(1-\eta)\frac{\mu}{L}\right](\phi_k-\phi^*). \tag{5.18}$$

We note that when $h = 0$ and the iteration becomes (5.2), condition (5.17) reduces to the norm test (5.3) with ξ given by $\eta/(1-\eta)$.

The assumption on α_k in Theorem 5.3.1 is fairly standard. The key is inequality (5.17), which is the most general condition we have identified for ensuring linear convergence. However, it does not seem to be possible to enforce this condition in practice, even approximately, for the composite optimization problem (which includes convex constrained optimization). Therefore, we seek an implementable version of (5.17), even if it is more restrictive. Before doing so, we show that condition (5.17) can also be used to establish convergence in the case when f is convex, but not strongly convex.

5.3.2. Convergence for General Convex f

We now show that when f is convex, the sequence of function values $\{\phi(x_k)\}$ converges to the optimal value ϕ^* of problem (5.6) at a sublinear rate, in expectation. To establish this result, we make the following assumptions.

Assumption 5.3.2. *$f : \mathbb{R}^n \to \mathbb{R}$ is convex, differentiable and has an $L-$ Lipschitz continuous gradient, and $h : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is a closed, convex and proper function.*

We begin by proving a technical lemma.

Lemma 5.3.2. *Suppose that Assumptions* (5.3.2) *hold and that $\{x_k\}$ is generated by iteration* (5.7)*, where g_k satisfies* (5.14) *and $\alpha_k = (1-\eta)/L$ for $\eta \in (0,1)$. If in addition condition* (5.17) *is satisfied, we have that for any given $z \in \mathbb{T}_k$,*

$$\mathbb{E}_k\left[\phi_{k+1}\right] \leq \phi(z) + \frac{1}{\alpha_k}\mathbb{E}_k\left[(x_k - x_{k+1})^T(x_k - z)\right] - \frac{1}{2\alpha_k}\mathbb{E}_k\left[\|x_k - x_{k+1}\|^2\right]. \tag{5.19}$$

Proof. By Assumptions (5.3.2), we have that

$$\begin{aligned}
\phi_{k+1} &\leq f_k + \nabla f_k^T(x_{k+1} - x_k) + \frac{L}{2}\|x_{k+1} - x_k\|^2 + h_{k+1} \\
&\leq f(z) - \nabla f_k^T(z - x_k) + \nabla f_k^T(x_{k+1} - x_k) + \frac{L}{2}\|x_{k+1} - x_k\|^2 + h_{k+1} \quad \text{(by convexity of } f\text{)} \\
&\leq f(z) - \nabla f_k^T(z - x_k) + \nabla f_k^T(x_{k+1} - x_k) + \frac{L}{2}\|x_{k+1} - x_k\|^2 + h(z) \\
&\quad - \left(\frac{x_k - x_{k+1}}{\alpha_k} - g_k\right)^T (z - x_{k+1}) \quad \text{(by convexity of } h \text{ and definition of } x_{k+1}\text{)} \\
&= \phi(z) - \nabla f_k^T(z - x_k) + \nabla f_k^T(x_{k+1} - x_k) + \frac{L}{2}\|x_{k+1} - x_k\|^2 \\
&\quad - \left(\frac{x_k - x_{k+1}}{\alpha_k} - g_k\right)^T (z - x_k + x_k - x_{k+1}) \\
&= \phi(z) + (g_k - \nabla f_k)^T(z - x_k) + \frac{1}{\alpha_k}(x_k - x_{k+1})^T(x_k - z) \\
&\quad + \left(\frac{L}{2} - \frac{1}{\alpha_k}\right)\|x_k - x_{k+1}\|^2 + (\nabla f_k - g_k)^T(x_{k+1} - x_k), \quad \text{(rearraging terms)}
\end{aligned}$$

where the third inequality follows from the fact that $0 \in g_k + \partial h_{k+1} + \frac{x_k - x_{k+1}}{\alpha_k}$. Taking conditional expectation and using (5.17), we have

$$\begin{aligned}
\mathbb{E}_k\left[\phi_{k+1}\right] &\leq \phi(z) + \mathbb{E}_k\left[(g_k - \nabla f_k)^T(z - x_k)\right] + \frac{1}{\alpha_k}\mathbb{E}_k\left[(x_k - x_{k+1})^T(x_k - z)\right] \\
&\quad + \left(\frac{L}{2} - \frac{1}{\alpha_k}\right)\mathbb{E}_k\left[\|x_k - x_{k+1}\|^2\right] + \mathbb{E}_k\left[(\nabla f_k - g_k)^T(x_{k+1} - x_k)\right] \\
&= \phi(z) + \frac{1}{\alpha_k}\mathbb{E}_k\left[(x_k - x_{k+1})^T(x_k - z)\right] + \left(\frac{L}{2} - \frac{1}{\alpha_k}\right)\mathbb{E}_k\left[\|x_k - x_{k+1}\|^2\right] \\
&\quad + \mathbb{E}_k\left[(\nabla f_k - g_k)^T(x_{k+1} - x_k)\right] \quad \text{(since } z \in \mathbb{T}_k \text{)} \\
&\leq \phi(z) + \frac{1}{\alpha_k}\mathbb{E}_k\left[(x_k - x_{k+1})^T(x_k - z)\right] - \left(\frac{1}{\alpha_k} - \frac{L}{2} - \frac{\eta}{2\alpha_k}\right)\mathbb{E}_k\left[\|x_k - x_{k+1}\|^2\right] \quad \text{(by (5.17))} \\
&= \phi(z) + \frac{1}{\alpha_k}\mathbb{E}_k\left[(x_k - x_{k+1})^T(x_k - z)\right] - \frac{1}{2\alpha_k}\mathbb{E}_k\left[\|x_k - x_{k+1}\|^2\right],
\end{aligned}$$

where the last equality is due to $\alpha_k = \frac{1-\eta}{L}$. □

Theorem 5.3.2. *Suppose that Assumptions* (5.3.2) *hold and that* $\{x_k\}$ *is generated by iteration* (5.7)*, where* g_k *satisfies* (5.14) *and* $\alpha_k = \alpha = (1-\eta)/L$ *for* $\eta \in (0,1)$*. If in addition* (5.17) *is satisfied, we have*

$$\mathbb{E}\left[\phi_k - \phi^*\right] \leq \frac{L\|x_0 - x^*\|^2}{2(1-\eta)k}, \tag{5.20}$$

where x^* *is any optimal solution of problem* (5.6)*.*

Proof. From Lemma 5.3.2, for any $z \in \mathbb{T}_k$, we have

$$\mathbb{E}_k\left[\phi_{k+1}\right] \leq \phi(z) + \frac{1}{\alpha}\mathbb{E}_k\left[(x_k - x_{k+1})^T(x_k - z)\right] - \frac{1}{2\alpha}\mathbb{E}_k\left[\|x_k - x_{k+1}\|^2\right].$$

Now substituting $z = x^* \in \mathbb{T}_k$ and taking full expectations, we have

$$\begin{aligned}
\mathbb{E}\left[\phi_{k+1} - \phi^*\right] &\leq \frac{1}{\alpha}\mathbb{E}\left[(x_k - x_{k+1})^T(x_k - x^*)\right] - \frac{1}{2\alpha}\mathbb{E}\left[\|x_k - x_{k+1}\|^2\right] \\
&= \frac{1}{2\alpha}\mathbb{E}\left[2(x_k - x_{k+1})^T(x_k - x^*) - \|x_k - x_{k+1}\|^2\right] \\
&= \frac{1}{2\alpha}\mathbb{E}\left[\|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2\right].
\end{aligned}$$

Summing the above inequality for $k = 0$ to $k-1$, we get

$$\begin{aligned}
\frac{1}{k}\sum_{t=0}^{k-1}\mathbb{E}\left[\phi_{t+1} - \phi^*\right] &\leq \frac{1}{2\alpha k}\mathbb{E}\left[\|x_0 - x^*\|^2 - \|x_k - x^*\|^2\right] \\
&\leq \frac{\|x_0 - x^*\|^2}{2\alpha k}.
\end{aligned}$$

By substituting $z = x_k \in \mathbb{T}_k$ in Lemma 5.3.2, we have that the sequence of expected function values is a deceasing sequence; specifically,

$$\mathbb{E}_k \left[\phi_{k+1}\right] \leq \phi_k - \frac{1}{2\alpha}\mathbb{E}_k \left[\|x_k - x_{k+1}\|^2\right].$$

Therefore, we have,

$$\mathbb{E}\left[\phi_k - \phi^*\right] \leq \frac{1}{k}\sum_{t=0}^{k-1} \mathbb{E}\left[\phi_{t+1} - \phi^*\right] \leq \frac{\|x_0 - x^*\|^2}{2\alpha k}.$$

□

5.3.3. A Practical Condition

To obtain a condition that is more amenable to computation than (5.17), we look for an upper bound for the left hand side of this inequality and a lower bound for the right hand side. Imposing an inequality between these two bounds will imply (5.17).

Theorem 5.3.3. *Suppose that Assumptions 5.3.1 hold, that* $\{x_k\}$ *is generated by* (5.7) *with* $\alpha_k = (1-\eta)/L$ *for* $\eta \in (0,1)$*, and that* g_k *satisfies* (5.14)*. If* g_k *additionally satisfies*

$$Var_k\left[g_k\right] \leq \frac{\eta}{2}\left\|\frac{\mathbb{E}_k\left[x_{k+1}\right] - x_k}{\alpha_k}\right\|^2, \tag{5.21}$$

then (5.17) *holds and hence*

$$\mathbb{E}_k\left[\phi_{k+1} - \phi^*\right] \leq \left[1 - (1-\eta)\frac{\mu}{L}\right](\phi_k - \phi^*).$$

If instead of Assumptions 5.3.1, the weaker Assumptions 5.3.2 hold, then

$$\mathbb{E}\left[\phi_k - \phi^*\right] \leq \frac{L\|x_0 - x^*\|^2}{2(1-\eta)k},$$

where x^* is any optimal solution of problem (5.6).

Proof. We first note that in (5.17),

$$\begin{aligned}\mathbb{E}_k\left[\|x_{k+1} - x_k\|^2\right] &= \|\mathbb{E}_k\left[x_{k+1}\right] - x_k\|^2 + \mathrm{Var}_k\left[x_{k+1} - x_k\right] \\ &\geq \|\mathbb{E}_k\left[x_{k+1}\right] - x_k\|^2,\end{aligned}$$

which is a quantity that we can approximate with a sample estimation; as we argue in Section 5.3.4.

On the other hand, let

$$\hat{x}_{k+1} = \mathrm{prox}_{\alpha_k h}\left(x_k - \alpha_k \nabla f_k\right) \in \mathbb{T}_k.$$

Then, since the prox operator is a contraction mapping,

$$\begin{aligned}
&\mathbb{E}_k\left[(\nabla f_k - g_k)^T(x_{k+1} - x_k)\right] \\
&= \mathbb{E}_k\left[(\nabla f_k - g_k)^T(x_{k+1} - \hat{x}_{k+1})\right] + \mathbb{E}_k\left[(\nabla f_k - g_k)^T(\hat{x}_{k+1} - x_k)\right] \\
&= \mathbb{E}_k\left[(\nabla f_k - g_k)^T(x_{k+1} - \hat{x}_{k+1})\right] \quad (\text{since } \hat{x}_{k+1} \in \mathbb{T}_k) \\
&\leq \mathbb{E}_k\left[\|\nabla f_k - g_k\| \, \|x_{k+1} - \hat{x}_{k+1}\|\right] \\
&= \mathbb{E}_k\left[\|\nabla f_k - g_k\| \left\|\text{prox}_{\alpha_k h}\left(x_k - \alpha_k g_k\right) - \text{prox}_{\alpha_k h}\left(x_k - \alpha_k \nabla f_k\right)\right\|\right] \\
&\leq \alpha_k \mathbb{E}_k\left[\|\nabla f_k - g_k\|^2\right] \\
&= \alpha_k \text{Var}_k\left[g_k\right].
\end{aligned}$$

Thus, we have obtained both

$$\frac{\eta}{2}\|\mathbb{E}_k\left[x_{k+1}\right] - x_k\|^2 \leq \frac{\eta}{2}\mathbb{E}_k\left[\|x_{k+1} - x_k\|^2\right]$$

and

$$\alpha_k \mathbb{E}_k\left[(\nabla f_k - g_k)^T(x_{k+1} - x_k)\right] \leq \alpha_k^2 \text{Var}_k\left[g_k\right].$$

Therefore, if we require that

$$\text{Var}_k\left[g_k\right] \leq \frac{\eta}{2}\left\|\frac{\mathbb{E}_k\left[x_{k+1}\right] - x_k}{\alpha_k}\right\|^2,$$

it follows that condition (5.17) is satisfied. □

The significance of Theorem 5.3.3 is that it establishes the convergence of the algorithm under condition (5.21) which, although being more restrictive than condition (5.17), can

be approximated empirically, as shown in Section 5.3.4. Again, it is reassuring that when $h = 0$, condition (5.21) reduces to the norm test (5.3).

5.3.4. Choice of the Sample Size S_k

We now discuss how to ensure that condition (5.21) is satisfied. At iterate x_k, suppose we select S_k i.i.d. samples $\{\theta_0, \theta_1, \cdots, \theta_{S_k-1}\}$ from Θ, and set

$$g_k = \frac{1}{S_k} \sum_{i=0}^{S_k-1} \nabla_x F(x_k, \theta_i). \tag{5.22}$$

Clearly, (5.14) holds and the variance of g_k is given by

$$\mathrm{Var}_k [g_k] = \frac{\mathbb{E}_k [\|\nabla_x F(x_k, \theta) - \nabla f(x_k)\|^2]}{S_k}.$$

Therefore, (5.21) holds if S_k satisfies

$$\frac{\mathbb{E}_k [\|\nabla_x F(x_k, \theta) - \nabla f(x_k)\|^2]}{S_k} \leq \frac{\eta}{2} \left\| \frac{\mathbb{E}_k [x_{k+1}] - x_k}{\alpha_k} \right\|^2, \tag{5.23}$$

or

$$S_k \geq \mathbb{E}_k \left[\|\nabla_x F(x_k, \theta) - \nabla f(x_k)\|^2 \right] \Big/ \frac{\eta}{2} \left\| \frac{\mathbb{E}_k [x_{k+1}] - x_k}{\alpha_k} \right\|^2. \tag{5.24}$$

This is the theoretical condition suggested by our analysis. Based on this condition we can state

Corollary 5.3.1. *Suppose that Assumptions 5.3.1 hold, that $\{x_k\}$ is generated by (5.7) with $\alpha_k = (1 - \eta)/L$ for $\eta \in (0, 1)$ and with g_k given by (5.22). If the sample sizes S_k are chosen to satisfy (5.24) for all k, then (5.17) is satisfied and hence (5.18) holds.*

If instead of Assumptions 5.3.1, the weaker Assumptions 5.3.2 are satisfied, then (5.20) *holds.*

In practice, we need to estimate both quantities on the right hand side of (5.24) As was done e.g. in [**14, 16**] the population variance term in the numerator can be approximated by a sample average:

$$\mathbb{E}_k \left[\|\nabla_x F(x_k, \theta) - \nabla f(x_k)\|^2 \right] \approx \frac{1}{S-1} \sum_{i=0}^{S-1} \|\nabla_x F(x_k, \theta_i) - \overline{g}_k\|^2 , \tag{5.25}$$

where $\overline{g}_k$ is defined in (5.10). We handle the denominator on the right hand side of (5.24) by approximating $\|\mathbb{E}_k[x_{k+1}] - x_k\|$ with the norm of the trial step $\|\overline{x}_{k+1} - x_k\|$ defined in (5.11), i.e.,

$$\overline{x}_{k+1} = \operatorname{prox}_{\alpha_k h} \left(x_k - \alpha_k \overline{g}_k \right) .$$

This is somewhat analogous to the approximations made in the unconstrained case in [**14, 16**], the main difference being the presence here of the prox operator, which is a contraction. Given this, we can replace (5.24) by

$$S_k \geq \frac{1}{S-1} \sum_{i=0}^{S-1} \|\nabla_x F(x, \theta_i) - \overline{g}_k\|^2 \Bigg/ \frac{\eta}{2} \left\| \frac{\overline{x}_{k+1} - x_k}{\alpha_k} \right\|^2 . \tag{5.26}$$

Our algorithm will use condition (5.26) to control the sample size.

5.3.5. The Practical Adaptive Sampling Algorithm

We now summarize the proposed method in this chapter as Algorithm 5.2, which employs the aforementioned approximations.

Algorithm 5.2: *Complete Algorithm for Solving Problem* (5.6)

1: **Input**: x_0, initial sample size $S \in \mathbb{N}^+$, and sequence $\{\alpha_k > 0\}$.

2: **for** $k = 1, 2, \ldots,$ **do**

3: Draw S i.i.d. samples $\{\theta_0, \theta_1, \cdots, \theta_{S-1}\}$ from Θ, compute

$$\overline{g}_k = \frac{1}{S} \sum_{i=0}^{S-1} \nabla_x F(x, \theta_i),$$

and trial proximal gradient step

$$\overline{x}_{k+1} = \text{prox}_{\alpha_k h} \left(x_k - \alpha_k \overline{g}_k \right). \tag{5.27}$$

4: Set

$$S_k = \max \{S, a\}, \; a = \frac{1}{S-1} \sum_{i=0}^{S-1} \|\nabla_x F(x, \theta_i) - \overline{g}_k\|^2 \Big/ \frac{\eta}{2} \left\| \frac{\overline{x}_{k+1} - x_k}{\alpha_k} \right\|^2. \tag{5.28}$$

5: **if** $S_k > S$ **then**

6: choose $(S_k - S)$ additional i.i.d. samples $\{\theta_S, \theta_{S+1}, \cdots, \theta_{S_k - 1}\}$ from Θ, and compute:

$$x_{k+1} = \text{prox}_{\alpha_k h} \left(x_k - \alpha_k g_k \right), \text{ where } g_k = \frac{1}{S_k} \sum_{i=0}^{S_k - 1} \nabla_x F(x, \theta_i);$$

7: **else**

8: set $x_{k+1} = \overline{x}_{k+1}$.

9: **end if**

10: Set $S = S_k$.

11: **end for**

As noted above, for large S, the choice of S_k given by (5.28) approximately ensures the condition (5.21) is satisfied. Computing S_k involves one evaluation of the proximal operator as well as the evaluation S stochastic gradients.

5.4. Using an Inner-Product Test in Place of the Norm Test

In the unconstrained setting, Bollapragada et al [**14**] have observed that the norm test, although endowed with optimal theoretical convergence rates, is too demanding in terms of sample size requirements. They derived a practical test called the *inner-product test*, which ensures that the search directions are descent directions with high probability, and performs well with smaller sample sizes. In this section, we extend these ideas to the constrained (or composite) optimization settings, and derive the equivalent inner-product test for adaptively controlling the sample sizes.

The goal is to choose the sample sizes such that the algorithm step provides descent with high probability. We would like to choose a sample size so that $\bar{d}_k = (\bar{x}_{k+1} - x_k)/\alpha_k$ provides decrease on the objective approximation $\nabla f(x_k)^T d + h(x_k + d)$, and specifically so that $\nabla f(x_k)^T \bar{d}_k + h(x_k + \bar{d}_k) - h(x_k) \leq \beta(\bar{g_k}^T \bar{d}_k + h(x_k + \bar{d}_k) - h(x_k))$ for some $\beta \in (0, 1)$. This means we want to satisfy

$$(\nabla f(x_k) - \bar{g_k})^T \bar{d}_k \leq -(1 - \beta) \left(\bar{g_k}^T \bar{d}_k + h(x_k + \bar{d}_k) - h(x_k) \right). \tag{5.29}$$

To estimate the left hand side of (5.29), note that in general, given a vector $p \in \mathbb{R}^n$, since $\mathbb{E}_k \left[(\bar{g} - \nabla f(x_k))^T p \right] = 0$, we can estimate the size of the quantity $(\bar{g_k} - \nabla f(x_k))^T p$ by estimating the variance of $(\bar{g_k} - \nabla f(x_k))^T p$. Since the initial sample size is S this is given by

$$\mathrm{Var}_k \left[\bar{g}_k^T p \right] \approx \frac{1}{S_k} \frac{1}{S-1} \sum_{i=0}^{S-1} \left((\nabla_x F(x, \theta_i) - \bar{g}_k)^T p \right)^2. \tag{5.30}$$

We use this estimate in (5.29) with $p = \bar{d}_k$ and get the condition

$$(5.31)\quad S_k \geq \frac{1}{S-1}\sum_{i=0}^{S-1}\left((\nabla_x F(x,\theta_i) - \overline{g}_k)^T \bar{d}_k\right)^2 \Big/ (1-\beta)^2(\bar{g_k}^T \bar{d}_k + h(x_k + \bar{d}_k) - h(x_k))^2,$$

where $(1-\beta)^2$ is analogous to $\eta/2$ in (5.26). We are aware that, in using (5.30) with $p = \bar{d}_k$, we are treating $\bar{g_k}$ and $\bar{d}_k$ as independent while they are not, but the practical success of the inner product test in [**14**] indicates that this approach is worthy of exploration.

We also note that in a problem with convex constraints, where $h(\cdot)$ involves a convex indicator function with possibly infinite values, the algorithm will only generate feasible points x_k and $x_k + \bar{d}_k$, so that in the above discussion h only takes on finite values at those points.

The version of our algorithm using this approach consists of following Algorithm 5.2 with the right hand side of (5.31) used in place of the formula for a in step 4 in Algorithm 5.2, which is presented in 5.3.

Algorithm 5.3: *Algorithm for Solving Problem* (5.6) *using inner-product test*

Input: x_0, initial sample size $S \in \mathbb{N}^+$, and sequence $\{\alpha_k > 0\}$.

for $k = 1, 2, \ldots,$ **do**

Draw S i.i.d. samples $\{\theta_0, \theta_1, \cdots, \theta_{S-1}\}$ from Θ, compute

$$\overline{g}_k = \frac{1}{S} \sum_{i=0}^{S-1} \nabla_x F(x, \theta_i),$$

and trial proximal gradient step

$$\overline{x}_{k+1} = \text{prox}_{\alpha_k h} \left(x_k - \alpha_k \overline{g}_k \right).$$

Set $S_k = \max\{S, a\}$, where

$$a = \frac{1}{S-1} \sum_{i=0}^{S-1} \left((\nabla_x F(x, \theta_i) - \overline{g}_k)^T \bar{d}_k \right)^2 \Big/ (1-\beta)^2 (\bar{g_k}^T \bar{d}_k + h(x_k + \bar{d}_k) - h(x_k))^2.$$

if $S_k > S$ **then**

choose $(S_k - S)$ additional i.i.d. samples $\{\theta_S, \theta_{S+1}, \cdots, \theta_{S_k - 1}\}$ from Θ, and compute:

$$x_{k+1} = \text{prox}_{\alpha_k h} \left(x_k - \alpha_k g_k \right), \text{ where } g_k = \frac{1}{S_k} \sum_{i=0}^{S_k - 1} \nabla_x F(x, \theta_i);$$

else

set $x_{k+1} = \overline{x}_{k+1}$.

end if

Set $S = S_k$.

end for

5.5. Numerical Experiments

We conducted numerical experiments to illustrate the performance of the proposed algorithms. We consider binary classification problems where the objective function is given by the logistic loss with ℓ_1-regularization:

$$\phi(x) = \frac{1}{N}\sum_{i=1}^{N} \log(1 + \exp(-y^i x^T z^i)) + \lambda \|x\|_1. \tag{5.32}$$

Here $\{(z^i, y^i),\ i = 1, \ldots, N\}$ are the input output data pairs, and the regularization parameter is chosen as $\lambda = 1/N$. This problem falls into the general category of minimizing composite optimization problems of the form (5.5), where $f(x) = \mathbb{E}\left[\log(1 + \exp(-y^i x^T z^i))\right]$, with expectation taken over a discrete uniform probability distribution defined on the dataset, and $h(x) = \lambda\|x\|_1$. We use the data sets listed in Table 5.1.

Data Set	Data Points N	Variables d	Reference
`covertype`	581012	54	[**12**]
`gisette`	6000	5000	[**33**]
`ijcnn`	35000	22	[**39**]
`MNIST`	60000	784	[**38**]
`mushrooms`	8124	112	[**39**]
`sido`	12678	4932	[**39**]

Table 5.1. Characteristics of the binary datasets used in the experiments.

We implemented three different variants of proximal stochastic gradient methods where the batch sizes are either: (1) continuously increased at a geometric rate (labelled `GEOMETRIC`), i.e.,

$$S_k = \left\lceil S_0 \left(1 + \gamma\right)^k \right\rceil; \tag{5.33}$$

where $\gamma > 0$ is a parameter that will be varied in the experiments; (2) adaptively chosen based on the norm test (5.26) as in Algorithm 5.2 (labelled `NORM`); or (3) adaptively chosen based on the inner-product test (5.31) as in Algorithm 5.3 (labelled `IP`). The steplength parameter α_k in each method is chosen as the number in the set $\{2^{-10}, 2^{-7}, \cdots, 2^{15}\}$ that leads to best performance. The initial sample size was set to $S_0 = 2$. The methods are terminated if $\|x_{k+1} - x_k\|/\alpha_k \leq 10^{-8}$ or if 100 epochs (passes through entire dataset) are performed. An approximation ϕ^* of the optimal function value was computed for each problem by running the deterministic proximal gradient method for 50,000 iterations.

Figures 5.2 and 5.3 report the performance of the three methods for the dataset `mushroom`, for various values of the parameter γ in (5.33) and η in (5.26) and (5.31) (η in (5.31) corresponds to $2(1-\beta)^2$). Figure 5.2, the vertical axis measures the optimality gap, $\phi(x) - \phi^*$, and the horizontal axis measures the number of effective gradient evaluations, defined as $\sum_{j=0}^{k} S_j/N$. In Figure 5.3, the vertical axis measures the batchsize as a fraction of total number of data points N, and the horizontal axis measures the number of iterations. The results for other datasets in Table 5.1 can be found in Appendix A.

We observe that the inner product test is the most efficient in terms of effective gradient evaluations, which is indicative of the total computational work and CPU time. For the geometric strategy, smaller values of γ typically lead to better performance, as they prevent the batch size from growing too rapidly. The norm test gives a performance comparable to the best runs of the geometric strategy, but the latter has much higher variability. In other words, whereas the geometric strategy can be quite sensitive to the choice of γ, the norm and inner product tests are fairly insensitive to the choice of η.

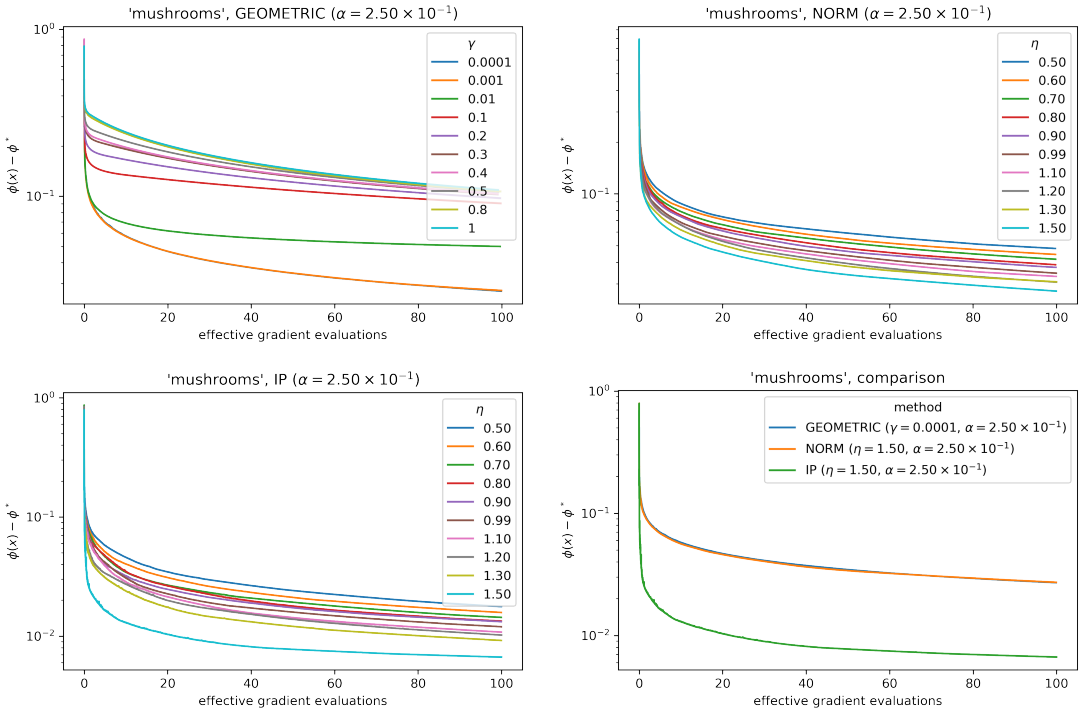

Figure 5.2. Optimality gap $\phi(x_k) - \phi^*$ against effective gradient evaluations on dataset `mushrooms`, with different strategies to control batch size: geometric increase (top left), norm test (top right), inner-product test (bottom left), and comparison between the best run for each method (bottom right).

5.6. Final Remarks

Algorithms that adaptively improve the quality of the approximate gradient during the optimization process are of interest from theoretical and practical perspectives, and have been well-studied in the context of unconstrained optimization. In this chapter, we proposed an adaptive method for solving constrained and composite optimization problems. The cornerstone of the proposed algorithm and its analysis is condition (5.17), which we regard as a natural generalization of the well-known norm test from unconstrained optimization. As this condition is difficult to implement precisely in practice, we approximate it by condition (5.21). We are able to prove convergence for the resulting

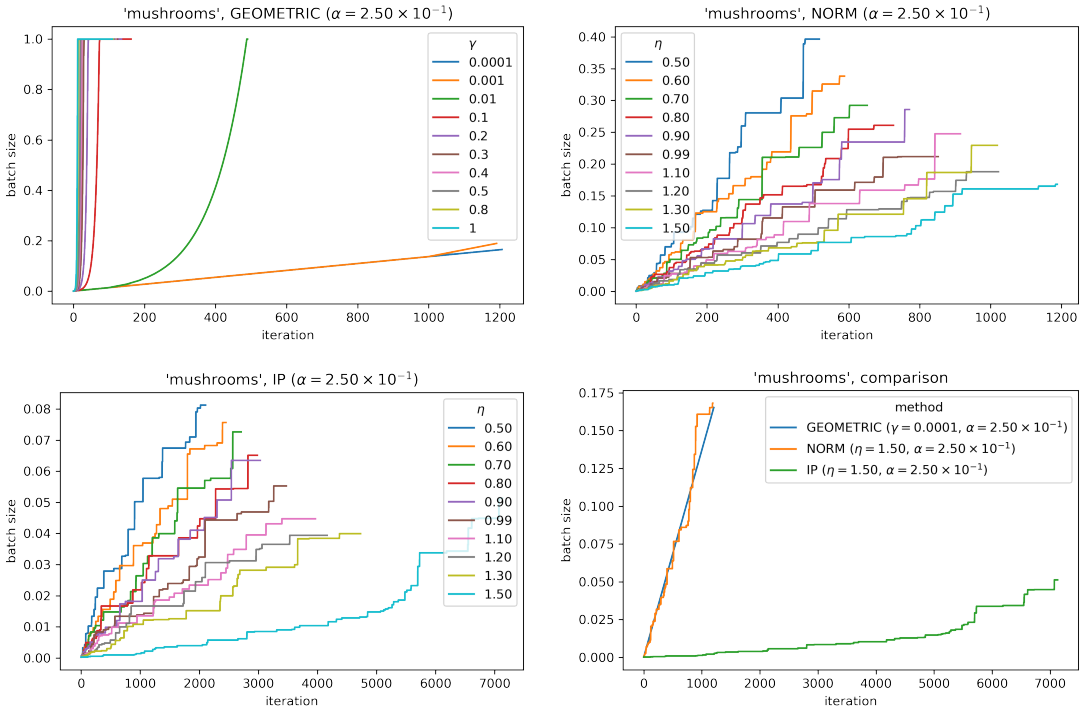

Figure 5.3. Batch size (as a fraction of total number of data points N) against iterations on dataset `mushrooms`, with different strategies to control batch size: geometric increase (top left), norm test (top right), inner-product test (bottom left), and comparison between the best run for each method (bottom right).

methods under standard conditions. It remains to be seen whether there is a condition with similar properties as (5.17), that is amenable to computation and less restrictive than (5.21). In this chapter, we also proposed a practical inner-product condition (5.29) that extends the ideas proposed in the unconstrained settings, and is more efficient in practice than the norm condition.

References

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

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

[3] F. Beiser, B. Keith, S. Urbainczyk, and B. Wohlmuth, *Adaptive sampling strategies for risk-averse stochastic optimization with constraints*, arXiv preprint arXiv:2012.03844, (2020).

[4] A. S. Berahas, R. H. Byrd, and J. Nocedal, *Derivative-free optimization of noisy functions via quasi-Newton methods*, arXiv preprint arXiv:1803.10173, (2018).

[5] ——, *Derivative-free optimization of noisy functions via quasi-newton methods*, SIAM Journal on Optimization, 29 (2019), pp. 965–993.

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

[7] ——, *A theoretical and empirical comparison of gradient approximations in derivative-free optimization*, arXiv preprint arXiv:1905.01332, (2019).

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

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

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

[11] D. P. Bertsekas, A. Nedić, and A. E. Ozdaglar, *Convex analysis and optimization*, Athena Scientific Belmont, 2003.

[12] J. A. Blackard and D. J. Dean, *Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables*, Computers and electronics in agriculture, 24 (1999), pp. 131–151.

[13] R. Bollapragada, R. Byrd, and J. Nocedal, *Adaptive sampling strategies for stochastic optimization*, arXiv preprint arXiv:1710.11258, (2017).

[14] ——, *Adaptive sampling strategies for stochastic optimization*, SIAM Journal on Optimization, 28 (2018), pp. 3312–3343.

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

[16] R. H. Byrd, G. M. Chin, J. Nocedal, and Y. Wu, *Sample size selection in optimization methods for machine learning*, Mathematical Programming, 134 (2012), pp. 127–155.

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

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

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

[20] R. G. Carter, *On the global convergence of trust region algorithms using inexact gradient information*, SIAM Journal on Numerical Analysis, 28 (1991), pp. 251–265.

[21] C. Cartis and K. Scheinberg, *Global convergence rate analysis of unconstrained optimization methods based on probabilistic models*, Mathematical Programming, (2015), pp. 1–39.

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

[23] C. Courtney Paquette and K. Scheinberg, *A stochastic line search method with convergence rate analysis*, arXiv preprint arXiv:1807.07994, (2018).

[24] Y.-H. Dai, *Convergence properties of the bfgs algoritm*, SIAM Journal on Optimization, 13 (2002), pp. 693–701.

[25] S. De, A. Yadav, D. Jacobs, and T. Goldstein, *Automated inference with adaptive batches*, in Artificial Intelligence and Statistics, 2017, pp. 1504–1513.

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

[27] M. P. Friedlander and M. Schmidt, *Hybrid deterministic-stochastic methods for data fitting*, SIAM Journal on Scientific Computing, 34 (2012), pp. A1380–A1405.

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

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

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

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

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

[33] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, *Result analysis of the NIPS 2003 feature selection challenge*, in Advances in neural information processing systems, 2004, pp. 545–552.

[34] R. W. Hamming, *Introduction to Applied Numerical Analysis*, Courier Corporation, 2012.

[35] A. Jalilzadeh, U. V. Shanbhag, J. H. Blanchet, and P. W. Glynn, *Optimal smoothed variable sample-size accelerated proximal methods for structured nonsmooth stochastic convex programs*, arXiv preprint arXiv:1803.00718, (2018).

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

[37] J. Larson, M. Menickelly, and S. M. Wild, *Derivative-free optimization methods*, Acta Numerica, 28 (2019), pp. 287–404.

[38] Y. LeCun, C. Cortes, and C. J. Burges, *MNIST handwritten digit database*, AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist, (2010).

[39] M. Lichman, *UCI machine learning repository*. `http://archive.ics.uci.edu/ml`, 2013.

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

[41] W. F. Mascarenhas, *The bfgs method with exact line searches fails for non-convex objective functions*, Mathematical Programming, 99 (2004), pp. 49–61.

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

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

[44] ——, *Estimating derivatives of noisy simulations*, ACM Transactions on Mathematical Software (TOMS), 38 (2012), p. 19.

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

[46] A. Nedić and D. Bertsekas, *Convergence rate of incremental subgradient algorithms*, in Stochastic optimization: algorithms and applications, Springer, 2001, pp. 223–264.

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

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

[49] R. Pasupathy, P. Glynn, S. Ghosh, and F. S. Hashemi, *On sampling rates in stochastic recursions*, (2015). Under Review.

[50] M. Powell, *Some global convergence properties of a variable metric algorithm for minimization without exact line searches*, in Nonlinear Programming, R. Cottle and C. Lemke, eds., Philadelphia, 1976, SIAM-AMS.

[51] M. Powell, *Some global convergence properties of a variable metric algorithm for minimization without exact line searches-nonlinear programming, vol. 4, siam-ams proceedings*, SIAM, Philadelpha, PA, (1976).

[52] M. J. D. Powell, *A fast algorithm for nonlinearly constrained optimization calculations*, in Numerical Analysis, Dundee 1977, G. A. Watson, ed., no. 630 in Lecture Notes in Mathematics, Heidelberg, Berlin, New York, 1978, Springer Verlag, pp. 144–157.

[53] F. Roosta-Khorasani and M. W. Mahoney, *Sub-sampled Newton methods I: Globally convergent algorithms*, arXiv preprint arXiv:1601.04737, (2016).

[54] ——, *Sub-sampled Newton methods II: Local convergence rates*, arXiv preprint arXiv:1601.04738, (2016).

[55] M. Schmidt, N. Roux, and F. Bach, *Convergence rates of inexact proximal-gradient methods for convex optimization*, in Advances in Neural Information Processing Systems, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, eds., vol. 24, Curran Associates, Inc., 2011, pp. 1458–1466.

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

[57] H.-J. M. Shi, M. Q. Xuan, F. Oztoprak, and J. Nocedal, *On the numerical performance of derivative-free optimization methods based on finite-difference approximations*, arXiv preprint arXiv:2102.09762, (2021).

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

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

APPENDIX A

Additional Numerical Experiments for Chapter 5

Here we present the numerical experiments for remaining data sets.

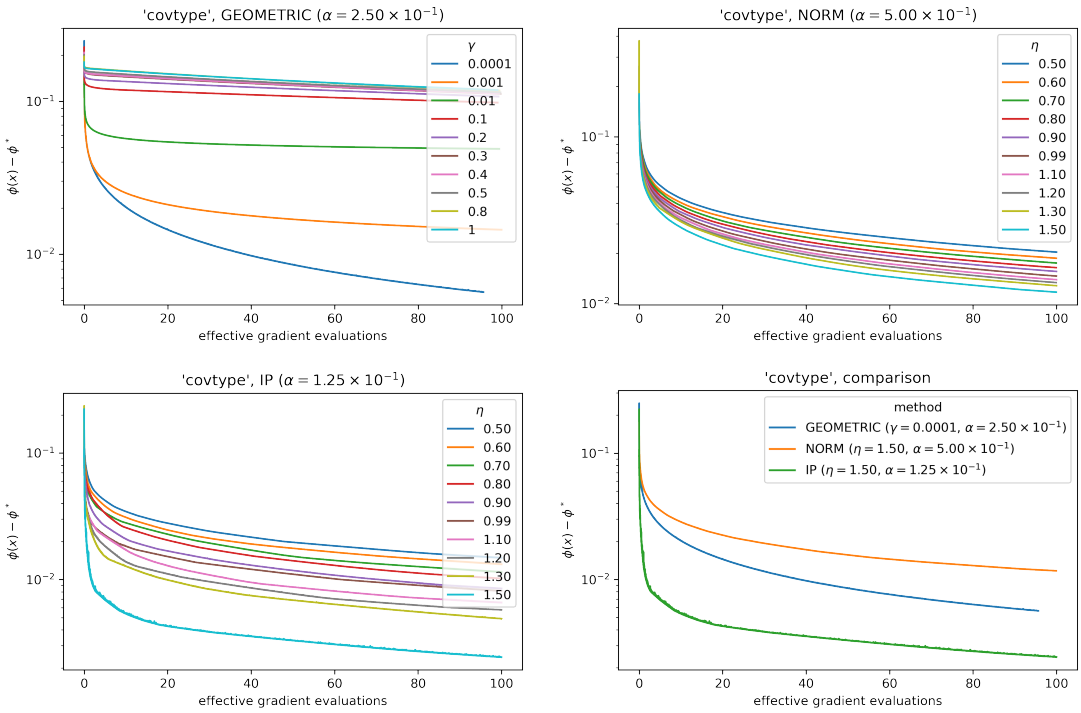

Figure A.1. Optimality gap $\phi(x_k) - \phi^*$ against effective gradient evaluations on dataset `covtype`, with different strategies to control batch size: geometric increase (top left), norm test (top right), inner-product test (bottom left), and comparison between the best run for each method (bottom right).

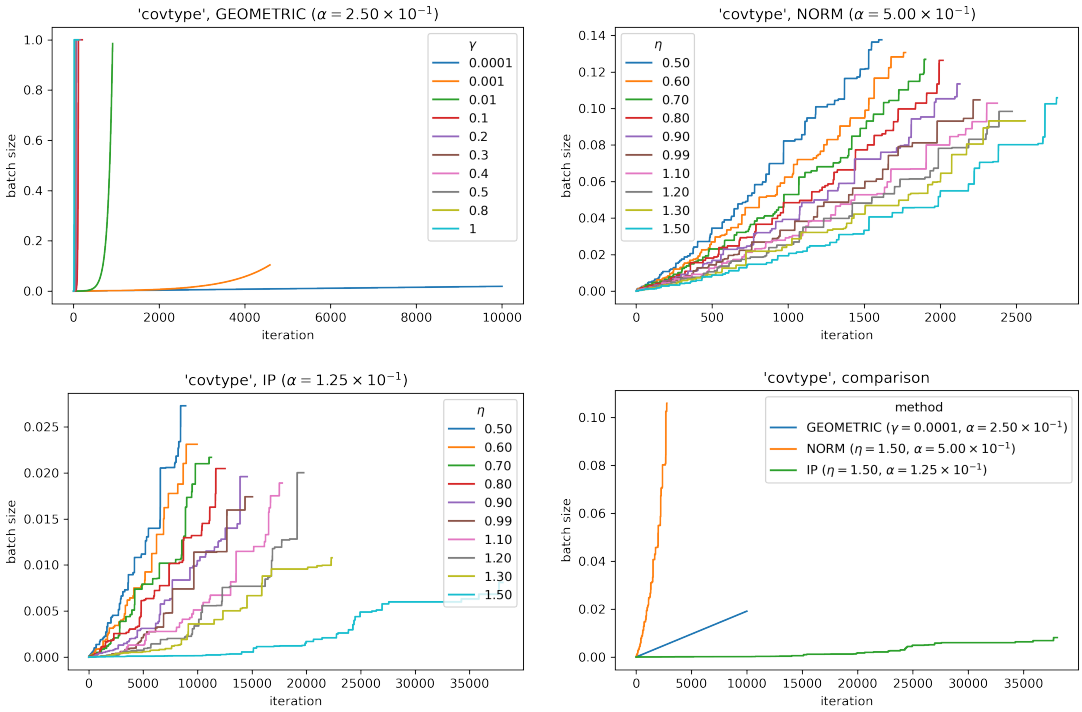

Figure A.2. Batch size (as a fraction of total number of data points N) against iterations on dataset `covtype`, with different strategies to control batch size: geometric increase (top left), norm test (top right), inner-product test (bottom left), and comparison between the best run for each method (bottom right).

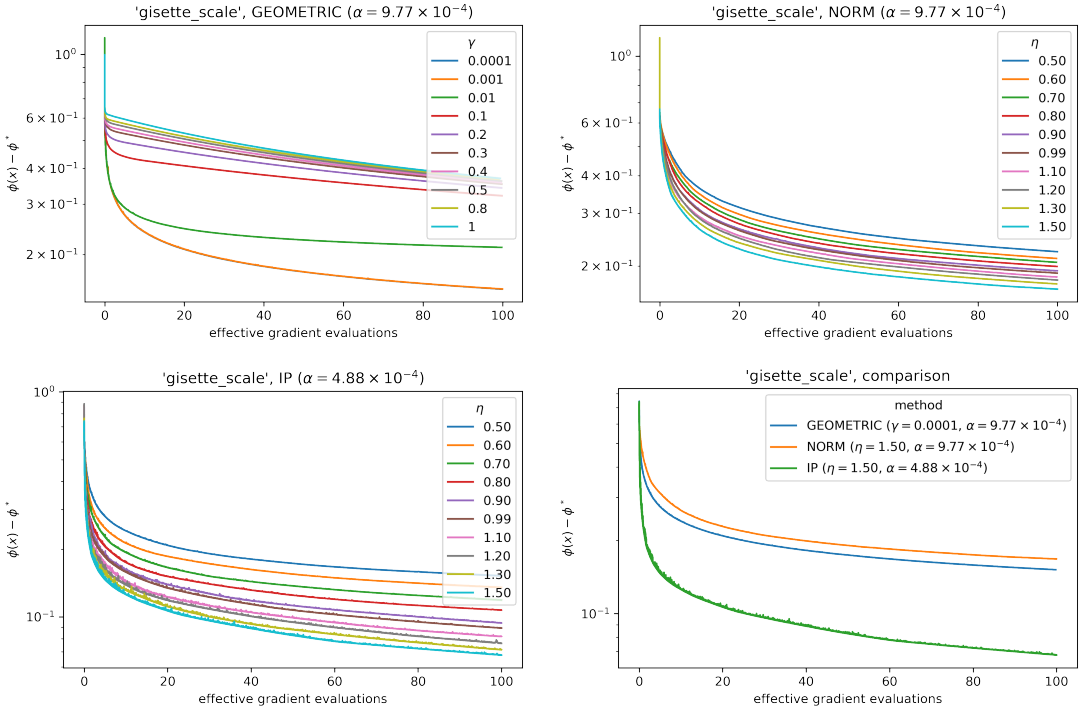

Figure A.3. Optimality gap $\phi(x_k) - \phi^*$ against effective gradient evaluations on dataset `gisette_scale`, with different strategies to control batch size: geometric increase (top left), norm test (top right), inner-product test (bottom left), and comparison between the best run for each method (bottom right).

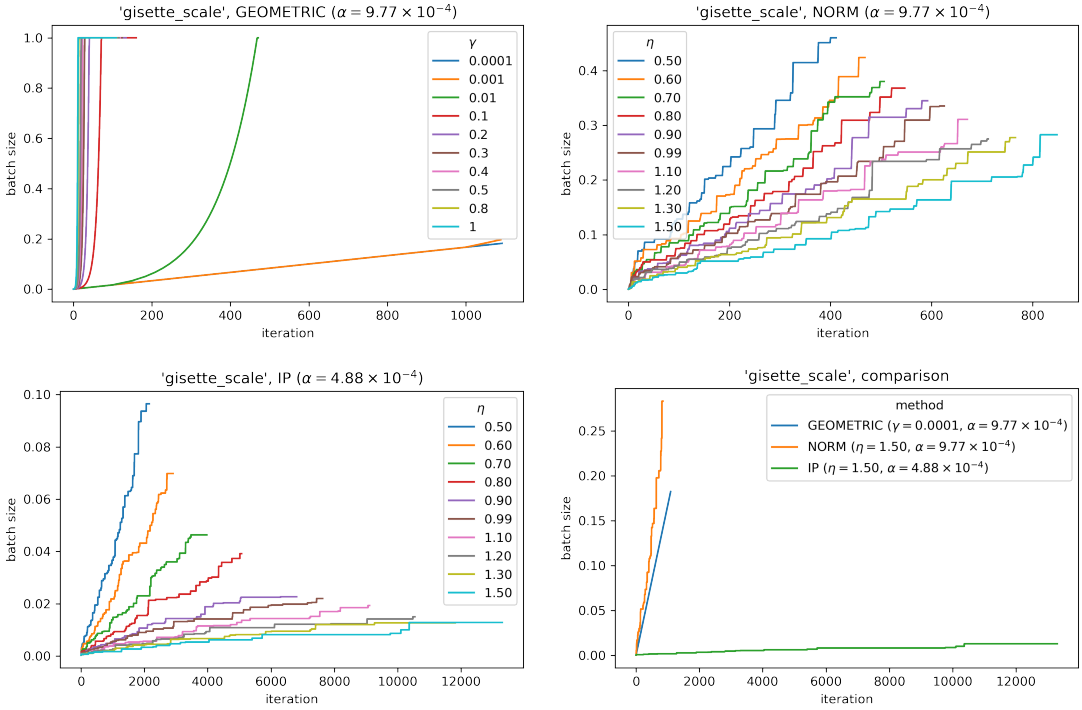

Figure A.4. Batch size (as a fraction of total number of data points N) against iterations on dataset `gisette_scale`, with different strategies to control batch size: geometric increase (top left), norm test (top right), inner-product test (bottom left), and comparison between the best run for each method (bottom right).

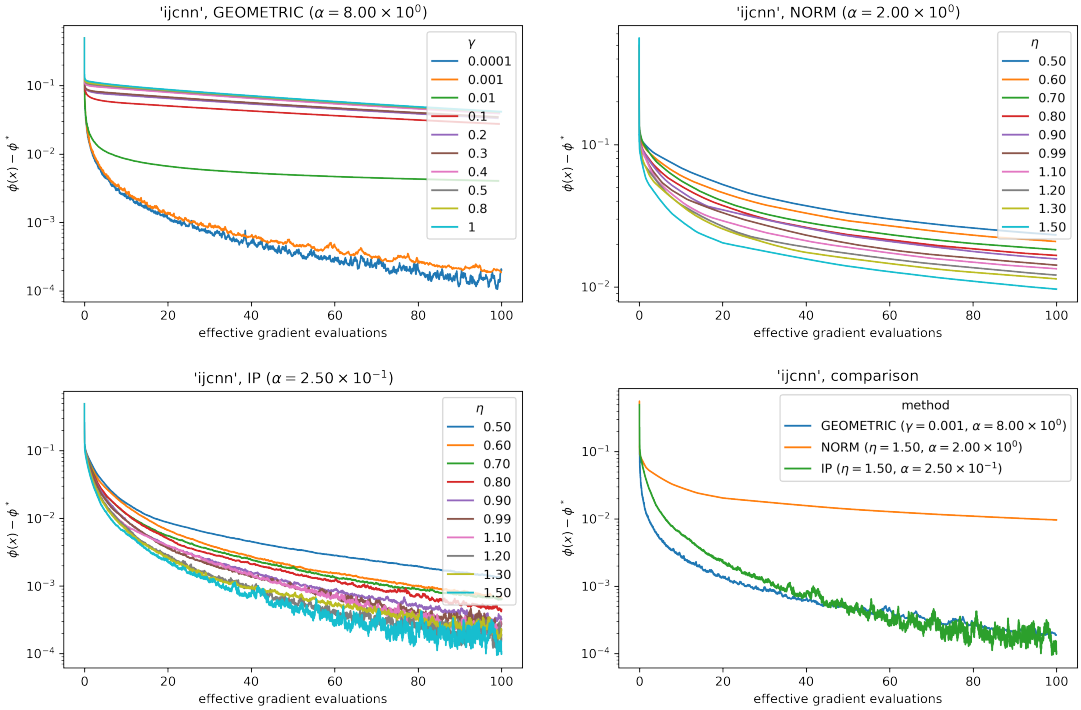

Figure A.5. Optimality gap $\phi(x_k) - \phi^*$ against effective gradient evaluations on dataset `ijcnn`, with different strategies to control batch size: geometric increase (top left), norm test (top right), inner-product test (bottom left), and comparison between the best run for each method (bottom right).

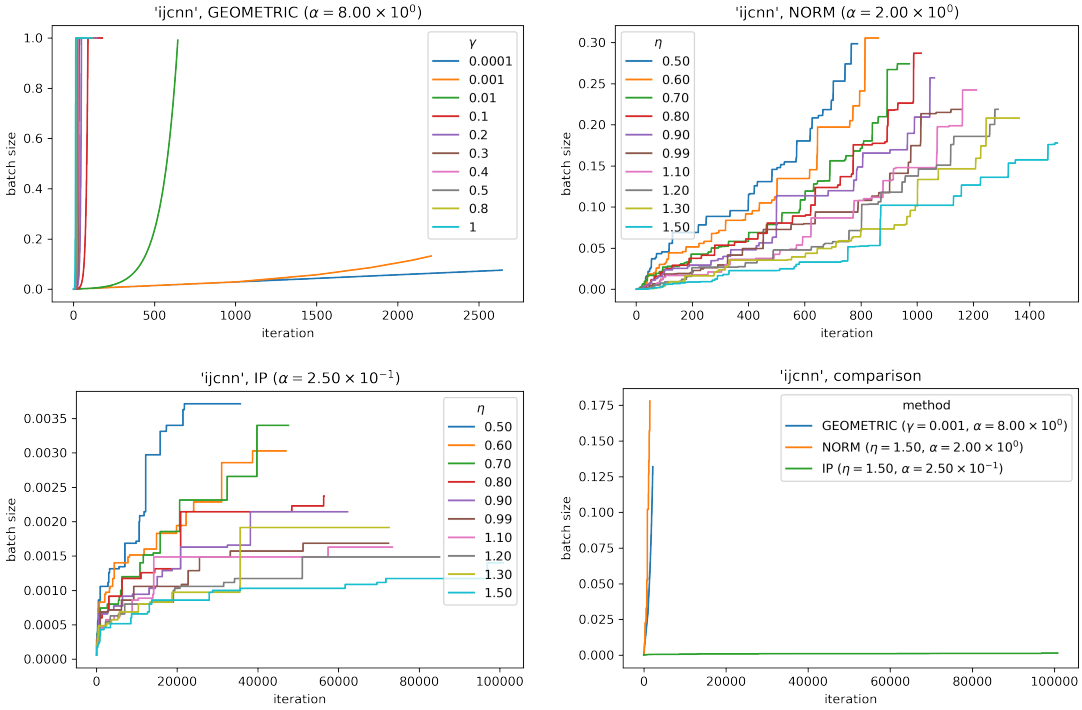

Figure A.6. Batch size (as a fraction of total number of data points N) against iterations on dataset `ijcnn`, with different strategies to control batch size: geometric increase (top left), norm test (top right), inner-product test (bottom left), and comparison between the best run for each method (bottom right).

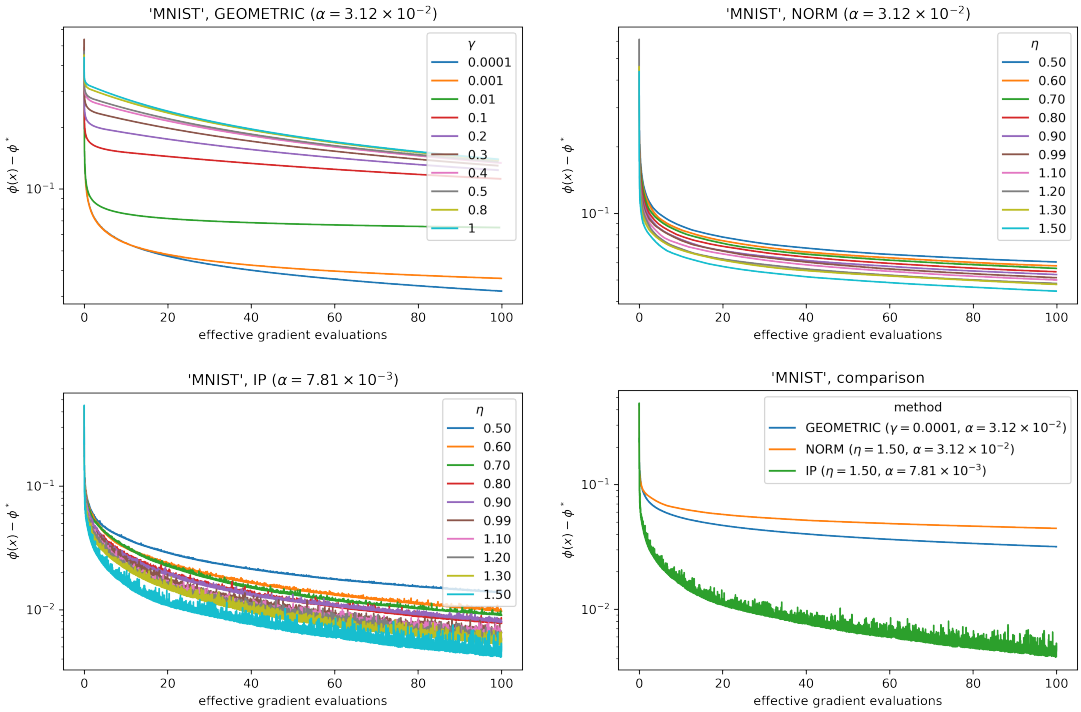

Figure A.7. Optimality gap $\phi(x_k) - \phi^*$ against effective gradient evaluations on dataset `MNIST`, with different strategies to control batch size: geometric increase (top left), norm test (top right), inner-product test (bottom left), and comparison between the best run for each method (bottom right).

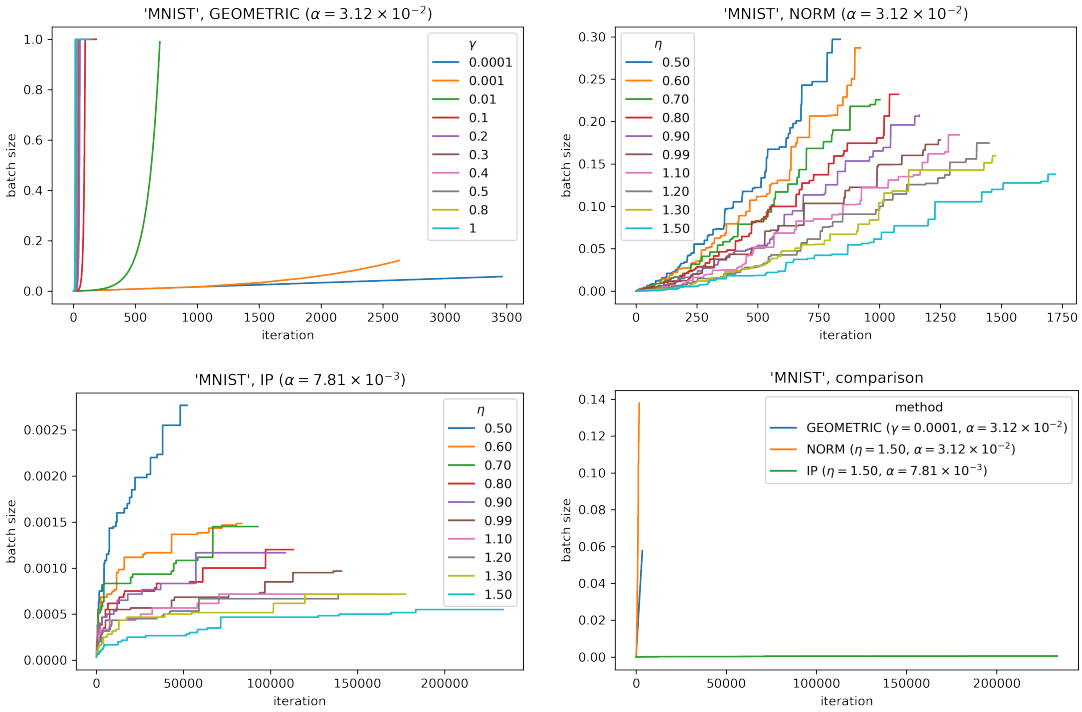

Figure A.8. Batch size (as a fraction of total number of data points N) against iterations on dataset `MNIST`, with different strategies to control batch size: geometric increase (top left), norm test (top right), inner-product test (bottom left), and comparison between the best run for each method (bottom right).

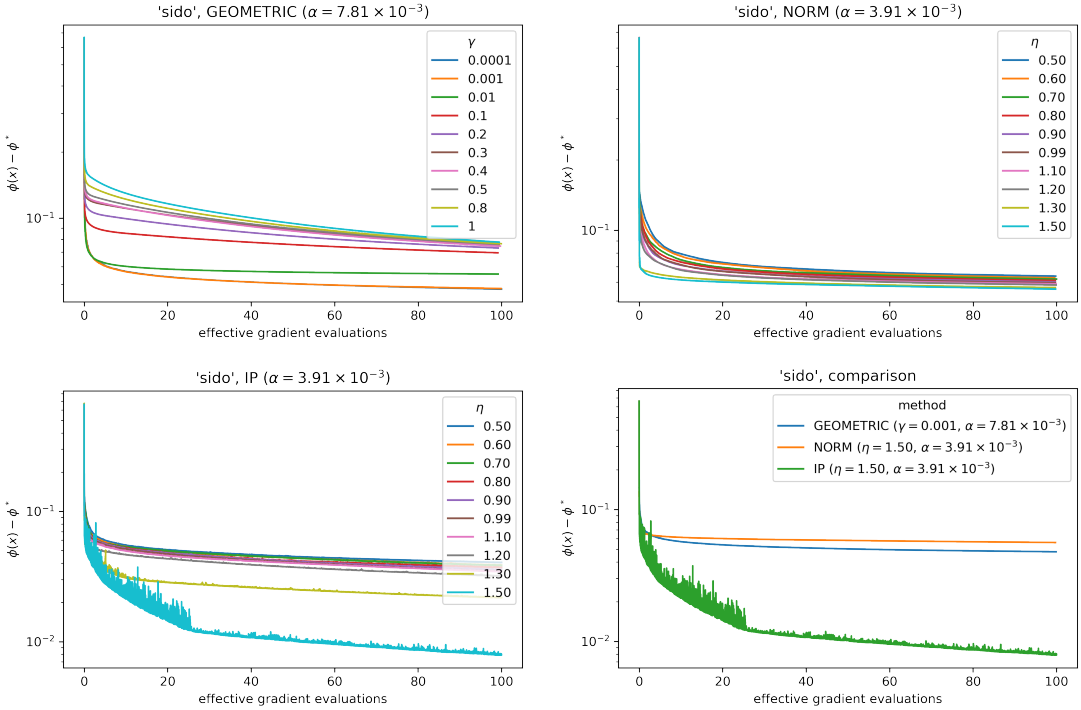

Figure A.9. Optimality gap $\phi(x_k) - \phi^*$ against effective gradient evaluations on dataset `sido`, with different strategies to control batch size: geometric increase (top left), norm test (top right), inner-product test (bottom left), and comparison between the best run for each method (bottom right).

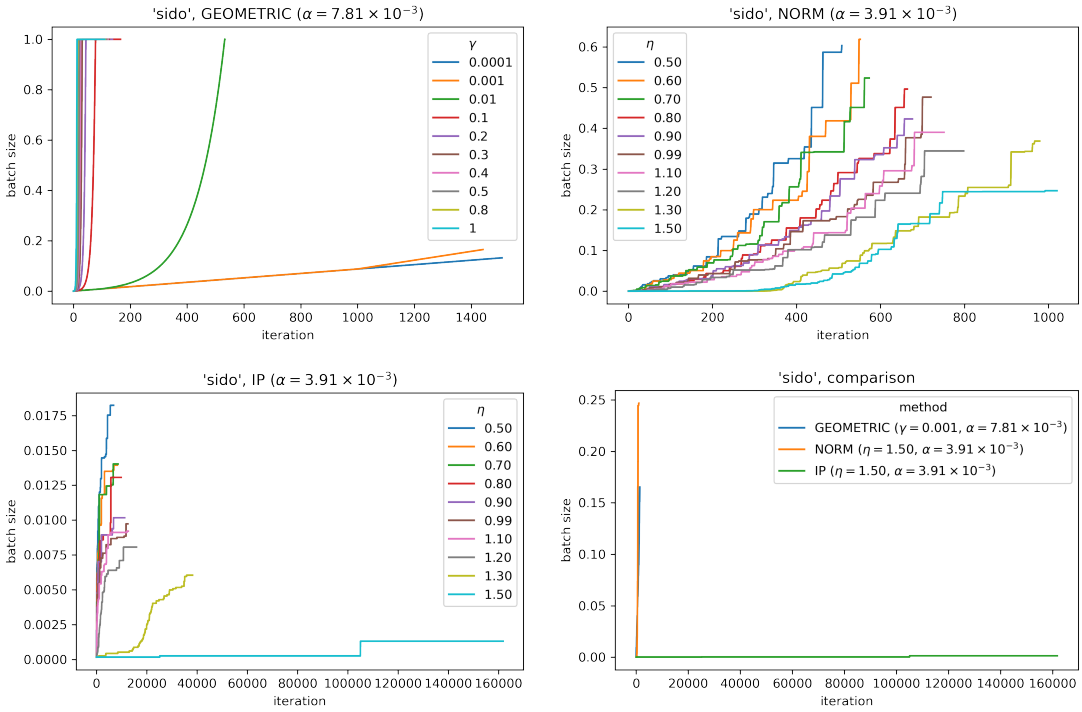

Figure A.10. Batch size (as a fraction of total number of data points N) against iterations on dataset `sido`, with different strategies to control batch size: geometric increase (top left), norm test (top right), inner-product test (bottom left), and comparison between the best run for each method (bottom right).