Lehigh University
Lehigh Preserve

Theses and Dissertations

2017

Random Models in Nonlinear Optimization

Matthew Joseph Menickelly
Lehigh University

Follow this and additional works at: https://preserve.lehigh.edu/etd

 Part of the Operations Research, Systems Engineering and Industrial Engineering Commons

Recommended Citation
Menickelly, Matthew Joseph, "Random Models in Nonlinear Optimization" (2017). *Theses and Dissertations*. 2957.
https://preserve.lehigh.edu/etd/2957

This Dissertation is brought to you for free and open access by Lehigh Preserve. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Lehigh Preserve. For more information, please contact preserve@lehigh.edu.

Random Models in Nonlinear Optimization

by

Matt Menickelly

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Doctor of Philosophy

in

Industrial and Systems Engineering

Lehigh University

August 2017

© Copyright by Matt Menickelly 2017

All Rights Reserved

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

_____
Date

_____
Dissertation Advisor

Committee Members:

_____
Katya Scheinberg, Committee Chair

_____
Frank E. Curtis

_____
Martin Takáč

_____
Stefan M. Wild

Acknowledgements

I am extremely grateful to Katya Scheinberg for her guidance and support as my Ph.D. advisor over the past five years, and I cannot sufficiently articulate how profound an impact she has had on my development as a mathematician. I am also particularly grateful to Stefan Wild and Jeff Larson, who served as mentors during two summers that I spent at Argonne National Laboratory, where I will be returning for a postdoctorate and, hopefully, great future collaborations. I thank the faculty of the Department of Industrial and Systems Engineering at Lehigh University, particularly Frank Curtis and Martin Takáč, who have served on my dissertation committee. I also thank Dzung Phan, who served as a mentor during my time at IBM T.J. Watson Research Center.

I thank the friends/future colleagues that I met at Lehigh, particularly Hiva, Reza, Pelin, Sertalp, Ali, Choat, and Wei. I'm not sure how I'll ever get work done again without the constant distractions provided by a proper subset of you all.

I would be remiss not to credit my husband, Andrew Palomo, who has provided me with a solid foundation in Bethlehem during the completion of my Ph.D., and has been a never-ending source of family, love, laughter, support, sanity, and motivation.

Finally, I am grateful to my family for providing me with the genetic material and nurturing environment necessary to become the Übermensch I am today. In all seriousness, I love my parents (Charlie and Betsy), my brother (Joe), and my sister (Molly), without whom I would never have had the support necessary to pursue a Ph.D, even if much of said support boiled down to constant reminders of a classic dialogue from *The Simpsons*:

BART: Look at me, I'm a grad student! I'm 30 years old and I made $600 last year!

MARGE: Bart, don't make fun of grad students. They just made a terrible life choice.

Contents

List of Tables

List of Figures

Abstract

In recent years, there has been a tremendous increase in the interest of applying techniques of deterministic optimization to stochastic settings, largely motivated by problems that come from machine learning domains. A natural question that arises in light of this interest is the extent to which iterative algorithms designed for deterministic (nonlinear, possibly non-convex) optimization must be modified in order to properly make use of inherently random information about a problem. This thesis is concerned with exactly this question, and adapts the model-based trust-region framework of derivative-free optimization (DFO) for use in situations where objective function values or the set of points selected by an algorithm to be objectively evaluated are random.

In the first part of this thesis, we consider an algorithmic framework called STORM (STochastic Optimization with Random Models), which as an iterative method, is essentially identical to model-based trust-region methods for smooth DFO. However, by imposing fairly general probabilistic conditions related to the concept of fully-linearity on objective function models and objective function estimates, we prove that iterates of algorithms in the STORM framework exhibit almost sure convergence to first-order stationary points for a broad class of unconstrained stochastic functions. We then show that algorithms in the STORM framework enjoy the canonical rate of convergence for unconstrained non-convex optimization. Throughout the thesis, examples are provided demonstrating how the mentioned probabilistic conditions might be satisfied through particular choices of model-building and function value estimation.

In the second part of the thesis, we consider a framework called manifold sampling, intended for unconstrained DFO problems where the objective is nonsmooth, but enough

is known a priori about the structure of the nonsmoothness that one can classify a given queried point as belonging to a certain smooth manifold of the objective surface. We particularly examine the case of sums of absolute values of (non-convex) black-box functions. Although we assume in this work that the individual black-box functions can be deterministically evaluated, we consider a variant of manifold sampling wherein random queries are made in each iteration to enhance the algorithm's "awareness" of the diversity of manifolds in a neighborhood of a current iterate. We then combine the ideas of STORM and manifold sampling to yield a practical algorithm intended for non-convex ℓ_1-regularized empirical risk minimization.

Chapter 1

Introduction

Derivative free optimization (DFO) [21] has grown in the past couple decades as a field of nonlinear optimization which addresses the optimization of black-box functions. By black-box functions, we mean functions whose zeroth-order value can be (approximately) computed through some numerical procedure or an experiment, but their closed-form expressions and derivatives are not available or else cannot be approximated accurately or efficiently. A reasonable question to ask is whether and when the machinery of derivative-free optimization, which assumes inherent error in the computation of first (and higher) order derivative information, can be applied to stochastic optimization [54, 67], where errors in derivative information are typically the result of stochasticity as opposed to numerical noise.

This thesis is concerned with addressing precisely this question. We will consider a well-studied trust-region DFO framework and extend it to consider a fairly general and adaptable framework, called STORM (STochastic Optimization with Random Models), which encompasses many typical situations of stochastic optimization. In particular, suppose there exists a deterministic "target" function $f : \mathbb{R}^n \to \mathbb{R}$ which we seek to minimize. For convenience, we will further suppose that this f has Lipschitz continuous gradient ∇f (although we do not assume ∇f is computable) and is bounded below. For whatever reason, e.g. deterministic noise arising from a discretization or an iterative process, or stochastic noise arising from a simulation or Monte Carlo sampling, we assume we cannot

compute $f(x)$ for a given $x \in \mathbb{R}^n$ directly, but instead only have access to a noisy observation $\tilde{f}(x, \xi(x))$ where $\xi(x)$ is a random variable with a distribution that may or may not be dependent on the argument x.

Continuing in this generality, we now suppose that there exist two certain machineries. The first machinery (model-building) accepts as input a probability α, a Euclidean ball centered at x^0 with radius $\delta > 0$ ($\mathcal{B}(x^0, \delta)$), and access to a set of input-observation pairs $\{(x, \tilde{f}(x, \xi(x)))\}$. In turn, this model-building machinery outputs a model m of f such that the worst-case error quantity

$$\max_{y \in \mathcal{B}(x^0, \delta)} |m(y) - f(y)|$$

is comparable to the error made by a linear Taylor model of f (i.e. $\mathcal{O}(\delta^2)$) with probability α. We can only require such a probabilistic bound to hold for this model-building machinery since we have not assumed anything more specific about the random variables $\xi(x)$

The second machinery (estimate-computation) accepts as input a probability β, a constant accuracy parameter ϵ_F, a parameter δ, two points $x^0, x^s \in \mathbb{R}^n$, and access to input-observation pairs $(x, \tilde{f}(x, \xi(x)))$. In turn, this second machinery outputs estimates $\bar{f}(x^0)$ and $\bar{f}(x^s)$ so that

$$|\bar{f}(x^0) - f(x^0)| < \epsilon_F \delta^2 \text{ and } |\bar{f}(x^s) - f(x^s)| < \epsilon_F \delta^2$$

hold simultaneously with probability β. The strength of STORM compared to many stochastic optimization methods from the point of view of analysis is that the failure of these two machineries, i.e. the generation of an arbitrarily poor-quality model or estimate, is occasionally permissible with fixed probability $1 - \alpha$ or $1 - \beta$, respectively.

The thesis is organized as follows. In Chapter 2, we will discuss the work in [17]. We will formalize the machineries just mentioned and describe the STORM algorithmic framework, which makes use of these machineries. Provided that model-building and estimate-computation machineries as described exist for the target function f and the random variables $\xi(x)$, we prove that there exists a choice of α, β, and ϵ_F so that the iterates of STORM will globally converge almost surely to a first-order stationary point of the tar-

get function f. We will provide some rudimentary examples of biased noise scenarios and demonstrate empirically that the method can be made to apply to scenarios of both biased and unbiased noise.

In Chapter 3, we will prove results about the theoretical convergence rate of STORM. Having established the almost sure convergence of STORM in Section 2, it is natural to question the worst-case iteration complexity required to reach an ϵ-accurate iterate X_k. In the non-convex case, given an $\epsilon > 0$, this means we are interested in an upper bound on the number of iterations of STORM required before an iterate X_k satisfies $\|\nabla f(X_k)\| \leq \epsilon$. As is typical in non-convex optimization, we will prove that the iteration complexity of STORM to encounter such an iterate is $\mathcal{O}(1/\epsilon^2)$ iterations. To acheive this, we show how STORM generates a stochastic process of iterates that can be coupled with (upper bounded by) a certain renewal-reward process. We then proceed to analyze that renewal-reward process to prove our result.

In Chapter 4, we consider a particular case of yielding a model-building machinery. Under the assumption of subgaussian (hence, unbiased) stochastic noise, we consider linear least-squares regression models and show how, as a function of the noise and sampling rates, such regression models satisfy Taylor-like error bounds on the trust region. We analyze different sampling methods for the design of the regression models. Most significantly, we prove a result that demonstrates how, with high probability, random uniform sampling in the trust region leads to models roughly as accurate as a forward-difference (or central-difference) gradient computed from averaged function values at the same sampling rate. Moreover, we demonstrate that the common theoretical tool in derivative-free optimization of using strongly well-poised designs (for instance, aggregating multiple random rotations of a forward-difference gradient) does not yield theoretical guarantees as strong as those from other sampling methods, as the assumption of a strongly well-poised set is quite general.

In Chapter 5, we will discuss an algorithm intended for the minimization of a sum of absolute values of black-box functions, which we call manifold sampling. Manifold sampling couples the DFO trust-region framework with a scheme similar to approximate gradient

sampling, but wholly different. The underlying assumption on the nonsmooth objective (which certainly holds true for ℓ_1 functions) is that the function is almost everywhere differentiable, and we can identify smooth manifolds in the immediate neighborhood of any given point based on the values returned by the black-box function being optimized. By building a smooth "master model", which incorporates gradient information from identified manifolds in a neighborhood of the current iterate on each iteration, the algorithm behaves like a smooth trust-region method. We prove that the algorithm iterates cluster around Clarke stationary points. The version of a manifold sampling algorithm that we will describe here can benefit from the use of random sampling (and thus, random models), and hence, although we will assume the black box functions can be computed without noise in Chapter 5, the fact that master models can be random fits into the scope of this thesis. We will suggest theoretically and demonstrate empirically that random models in this context have the capability to outperform deterministic models, particularly in terms of robustness.

In Chapter 6, we will propose a variant of STORM, coupled with the ideas of manifold sampling from Chapter 5, to yield an algorithm for solving an ℓ_1-regularized non-convex empirical risk minimization problem, a problem of interest in machine learning. We will demonstrate that in this problem setting, this variant of STORM, which we call HAIL-STORM, tends to yield classifiers at least as good as those found by well-tuned stochastic gradient descent methods in the same number of data passes, but with virtually no parameter tuning. Moreover, we demonstrate the importance of solving this problem in terms of robustness of classifiers to misclassification noise.

Lastly, in Chapter 7, we will provide some concluding remarks and directions for future work.

Chapter 2

STORM

2.1 Introduction

In this chapter[1], using methods developed for DFO, we aim to solve

$$\min_{x \in \mathbf{R}^n} \; f(x) \tag{2.1}$$

where $f(x)$ is a function which is assumed to be smooth and bounded from below, but the value of which can only be computed with some noise. Let $\tilde{f}$ be the noisy computable version of f, which takes the form

$$\tilde{f}(x) = f(x, \varepsilon),$$

where the noise ε is a random variable.

In recent years, some DFO methods have been extended to and analyzed for stochastic functions [24, 48]. Additionally, stochastic approximation methodologies started to incorporate techniques from the DFO literature [15]. The analysis in all that work assumes some particular structure of the noise, including the assumption that the noisy function values give an unbiased estimator of the true function value.

[1]The work in this Chapter is joint work with Ruobing Chen and Katya Scheinberg. A published version can be found in [17] and an earlier version can be found in the last chapter of Ruobing Chen's PhD thesis [16].

There are two main classes of methods in this setting of stochastic optimization: stochastic gradient (SG) methods (such as the well known Robbins-Monro method) and sample average approximation (SAA) methods. The former (SG) methods work roughly as follows: they obtain a realization of an unbiased estimator of the gradient at each iteration and take a step in the direction of the negative gradient. The step sizes progressively diminish and the iterates are averaged to form a sequence that converges to a solution. These methods typically have very inexpensive iterations, but exhibit slow convergence, with the convergence rate being strongly dependent on the choice of algorithmic parameters, particularly the sequence of step sizes. Many variants exist that average the gradient information from past iterations and are able to accept sufficiently small, but nondecreasing step sizes [3, 41]. SG methods have enjoyed extraordinary popularity with their application in the field of machine learning (e.g., see an extensive survey on the subject in [10]). Many sophisticated variants, that involve acceleration and other techniques have been developed and have been shown to be efficient in practice [33, 47, 56]. However, the majority of these methods aim exclusively at convex functions and may not converge in non-convex settings. Variance reduction techniques, such as in [23, 40], have been proposed for cases where $f(x)$ is a finite sum of convex functions, a much more restrictive setting than what we consider here. While some variants of the above methods exist for non-convex problems (e.g. [34]), the convergence remains slow, and parameter tuning remains necessary in most cases.

The second class of methods, SAA, is based on sample averaging of the function and gradient estimators, which is applied to reduce the variance of the noise. These methods repeatedly sample the function value at a set of points in hopes to ensure sufficient accuracy of the function and gradient estimates. For a thorough introduction and references therein, see [58]. The optimization method and sampling process are usually tightly connected in these approaches; hence, again, algorithmic parameters need to be specially chosen and tuned. SAA methods tend to be more robust with respect to parameters and enjoy faster convergence at a cost of more expensive iterations. Practical success has been demonstrated for specially designed methods for problems of particular structure (see, e.g. [52]). However, very few SAA methods have been developed specifically for trust region meth-

ods and general non-convex problems. Moreover, none of the methods mentioned above are applicable in the case of biased noise, and they suffer significantly in the presence of outliers.

We will show that a *standard, efficient, unconstrained optimization method* - in this case, a trust region method - can be applied, with very small modifications, to stochastic nonlinear (not necessarily convex) functions and can be guaranteed to converge to first-order stationary points as long as certain conditions are satisfied. We present a general framework, where we do not specify any particular sampling technique. The framework is based on the trust region DFO framework [21], and its extension to probabilistic models [4]. In terms of this framework and the certain conditions that must be satisfied, we essentially assume that

- the local models of the objective function constructed on each iteration satisfy some first-order accuracy requirement with sufficiently high probability,
- and that function estimates at the current iterate and at a potential next iterate are sufficiently accurate with sufficiently high probability.

The main novelty of this work is the analysis of the framework and the resulting weaker, more general, conditions for convergence compared to prior work. In particular,

- we do not assume that the probabilities of obtaining sufficiently accurate models and estimates are increasing (they simply need to be above a certain constant) and
- we do not assume any distribution of the random models and estimates. In other words, if a model or estimate is inaccurate, it can be arbitrarily inaccurate, i.e. the noise present in function evaluations can have nonconstant bias.

It is also important to note that while our framework and model requirements are borrowed from prior work in DFO, this framework applies to derivative-based optimization as well. Later in the chapter we will discuss different settings which will fit into the proposed framework.

This chapter consists of two main parts. In the first part, we propose and analyze a trust region framework, which utilizes random models of $f(x)$ at each iteration to compute the

next potential iterate. It also relies on (random, noisy) estimates of the function values at the current iterate and the potential iterate to gauge the progress that is being made. The convergence analysis then relies on requirements that these models and these estimates are sufficiently accurate with sufficiently high probability. Beyond these conditions, no assumptions are made about how these models and estimates are generated. The resulting method is a stochastic process that is analyzed with the help of martingale theory. The method is shown to converge to first-order stationary points with probability one.

In the second part of this chapter, we consider various scenarios under different assumptions on the noise-inducing component ε and discuss how sufficiently accurate random models can be generated. In particular, we show that in the case of unbiased noise, that is when $\mathbb{E}[f(x,\varepsilon)] = f(x)$ and $\mathrm{Var}[f(x,\varepsilon)] \leq \sigma^2 < \infty$ for all x, sample averaging techniques give us sufficiently accurate models. Although we will prove convergence under the discussed framework that essentially says we have the ability to compute both sufficiently accurate models and estimates with constant, separate probabilities, it is not necessarily easy to estimate what these probabilities ought to be for a given problem. While we provide some guidance on the selection of sampling rates in an unbiased noise setting in Section 5, our numerical experiments show that the bounds on probabilities suggested by our theory to be necessary for almost sure convergence are far from tight.

We also discuss the case where $\mathbb{E}[f(x,\varepsilon)] \neq f(x)$, and where the noise bias may depend on x or on the method of computation of the function values. One simple setting, which is illustrative, is as follows. Suppose we have an objective function, which is computed by a numerical process, whose accuracy can be controlled (for instance by tightening some stopping criterion within this numerical process). Suppose now that this numerical process involves some random component (such as sampling from a large dataset and/or utilizing a randomized algorithm). It may be known that with sufficiently high probability this numerical process produces a sufficiently accurate function value - however, with some small (but nonzero) probability the numerical process may fail and hence no reasonable value is guaranteed. Moreover, such failures may become more likely as more accurate computations are required (for instance because an upper bound on the total number of

iterations is reached inside the numerical process). Hence the probability of failure may depend on the current iterate and state of the algorithm. Here we simply assume that such failures do not occur with probability higher than some constant (which will be specified in our analysis), conditioned on the past. However, we do not assume anything about the magnitude of the inaccurate function values. As we will demonstrate later in this chapter, in this setting, $\mathbb{E}[f(x,\varepsilon)] \neq f(x)$.

2.2 Comparison with related work

There is a very large volume of work on SG and SAA, most of which is quite different from our proposed analysis and method. However, we will mention a few works here that are most closely related to this chapter and highlight the differences. The three methods existing in the literature we will compare with are by Deng and Ferris [24], SPSA (simultaneous perturbations stochastic approximation) [70, 71], and SCSR (sampling controlled stochastic recursion) [35]. These three settings and methods are most closely related to our work because they all rely on using models of the (possibly non-convex) objective function that can both incorporate second-order information and whose accuracy with respect to a "true" model can be dynamically adjusted. In particular, Deng and Ferris apply the trust-region model-based derivative free optimization method UOBYQA [60] in a setting of sample path optimization [64]. In [70] and [71], the author applies an approximate gradient descent and Newton method, respectively, with gradient and Hessian estimates computed from specially designed finite differencing techniques, with a decaying finite differencing parameter. In [35] a very general scheme is presented, where various deterministic optimization algorithms are generalized as stochastic counterparts, with the stochastic component arising from the stochasticity of the models and the resulting step of the optimization algorithm. We now compare some key components of these three methods with those of our framework, which we hereforth refer to as STORM (STochastic Optimization with Random Models).

Deng and Ferris: The assumptions of the sample path setting are roughly as follows: on each iteration k, given a collection of points $X^k = \{x_1^k, \ldots, x_p^k\}$ one can compute noisy function values $f(x_1^k, \varepsilon^k), \ldots, f(x_p^k, \varepsilon^k)$. The noisy function values are assumed to be re-

alizations of an unbiased estimator of true values $f(x_1^k), \ldots, f(x_p^k)$. Then, using multiple, say N_k, realizations of ε^k, average function values $f^{N_k}(x_1^k), \ldots, f^{N_k}(x_p^k)$ can be computed. A quadratic model $m_k^{N_k}(x)$ is then fit into these function values, and so a sequence of models $\{m_k^{N_k}(x)\}$ is created using a nondecreasing sequence of sampling rates $\{N_k\}$. The assumption on this sequence of models is that each of them satisfies a sufficient decrease condition (with respect to the true model of the true function f) with probability $1 - \alpha_k$, such that $\sum_{k=1}^{\infty} \alpha_k < \infty$. The trust region maintenance follows the usual scheme like that in UOBYQA, hence the step sizes taken by the algorithm can be increased or decreased depending on the observed improvement of the function estimates.

SPSA: The first-order version of this method assumes that $f(x, \varepsilon)$ is an unbiased estimate of $f(x)$, and the second-order version, 2SPSA, assumes that an unbiased estimate of $\nabla f(x)$, $g(x, \varepsilon) \in \mathbf{R}^n$, can be computed. Gradient (in the first-order case) and Hessian (in the second-order case) estimates are constructed using an interesting randomized finite differencing scheme. The finite difference step is assumed to be decaying. An approximate steepest descent direction or approximate Newton direction is then constructed and a step of length t_k is taken along this direction. The sequence $\{t_k\}$ is assumed to be decaying in the usual Robbins-Monro way, that is $t_k \to 0$, $\sum_k t_k = \infty$. Hence, while no increase in accuracy of the models is assumed (the models only need to be accurate in expectation), the step size parameter and the finite differencing parameter need to be tuned. Decaying step sizes often lead to slow convergence, as has been observed often in stochastic optimization literature.

SCSR: This is a very general scheme which can include multiple optimization methods and sampling rates. The key ingredients of this scheme are a deterministic optimization method, and a stochastic variant that approximates it. The stochastic step (in their work, *recursion*) is assumed to be a sufficiently accurate approximation of the deterministic step with increasing probability (the probabilities of failure in each iteration are summable). This assumption is stronger than the one in this chapter. In addition, another key assumption made for SCSR is that the iterates produced by the base deterministic algorithm converge to the unique optimal minimizer x^*. Not only do we not assume here that the

minimizer/stationary point is unique, but we also do not assume a priori that the iterates form a convergent sequence, since they may not do so in a non-convex setting.

STORM: Like the Deng and Ferris method, we utilize a trust-region, model-based framework, where the size of the trust region can be increased or decreased according to empirically observed function decrease and the size of the observed approximate gradients. The desired accuracy of the models is tied only to the trust-region radius in our case, while for Deng and Ferris, it is tied to both the radius and the size of true model gradients (the second condition is harder to ensure). In either method, this desired accuracy is assumed to hold with some probability - in STORM, this probability remains constant throughout the progress of the algorithm, while for Deng and Ferris, this probability must converge to 1 sufficiently rapidly.

There are three major advantages to our results. First of all, in the case of unbiased noise, the sampling rate is directly connected to the desired accuracy of the estimates and the probability with which this accuracy is achieved. Hence, for the STORM method, the sampling rate may increase or decrease according to the trust region radius, eventually increasing only when necessary, i.e. when the noise becomes dominating. For all the other methods listed here, the sampling rate is assumed to increase monotonically. Secondly, in the case of biased noise, we can still prove convergence of our method, as long as the desired accuracy is achieved with a fixed probability. In other words, we allow for noise to be arbitrarily dominating with a small, but fixed, probability on each iteration. This allows us to consider new models of noise which cannot be handled by any of the other methods discussed here. In addition, STORM incorporates first- and second-order models without changing the algorithm - the step size parameter (i.e., the trust region radius) and other parameters of the method are chosen similarly to the standard choices for parameters used in the trust-region methods, which have proven to be very effective in practice for unconstrained nonlinear optimization. In Section 2.7 we show that the STORM method is very effective in different noise settings and is very robust with respect to sampling strategies.

Finally, we point to [48], which proposes a very similar method to the one in this chapter.

Both methods were developed based on the trust-region DFO method with random models for deterministic functions analyzed in [4] and extended to the stochastic setting. Some of the assumptions in this chapter were inspired by an early version of [48]. However, the assumptions and the analysis in [48] are quite different from what appears in this chapter. In particular, they rely on the assumption that $f(x, \varepsilon)$ is an unbiased estimate of $f(x)$, hence their analysis does not extend to the biased case. Also they assume that the probability of having an accurate model at the k-th iteration is at least $1 - \alpha_k$, such that $\alpha_k \to 0$, while for our method this probability can remain bounded away from zero. Similarly, they assume that the probability of having accurate function estimates at the k-th iteration also converges to 1 sufficiently rapidly, while in our case it is again constant. Their analysis, as a result, is very different from ours, and does not generalize to various stochastic settings (they only focus on a derivative-free setting with additive noise). The advantage of their method is that they do not need to put a restriction on acceptable step sizes when the norm of the model gradient is small. We, on the other hand, impose such a restriction in our method and use it in the proof of our main result. However, as we discuss later in this chapter, this restriction can be relaxed at the cost of a more complex algorithm and analysis. In practice, we do not implement this restriction, hence our basic implementation is virtually identical to that in [48] except that we implement a variety of model-building strategies, while only one such strategy (regression models based on randomly rotated orthogonal samples sets) is implemented in [48] [2]. Thus we do not directly compare the empirical performance of our method with the method in [48] since we view it as more or less the same method.

We conclude this section by introducing some frequently used notations and their meanings. The rest of this chapter is organized as follows. In Section 2.3 we introduce the trust region framework, followed by Section 2.4, where we discuss the requirements on our random models and function estimates. The main convergence results are presented in Section 2.5. In Section 2.6, we discuss various noise scenarios and how sufficiently accurate models and estimates can be constructed in these cases. Finally, we present computational

[2] We will discuss this particular aspect of model-building in further detail in Chapter 4.

experiments based on these various noise scenarios in Section 2.7.

Notations. Let $\|\cdot\|$ denote the Euclidean norm and $B(x, \Delta)$ denote the ball of radius Δ around x, i.e., $B(x, \Delta) : \{y : \|x - y\| \leq \Delta\}$. Probability sample spaces are denoted by Ω, according to the context, and a sample from that space is denoted by $\omega \in \Omega$. As a rule, when we describe a random process within the algorithmic framework, uppercase letters, e.g. the k-th iterate X_k, will denote random variables, while lowercase letters will denote realizations of the random variable, e.g. $x_k = X_k(\omega)$ is the k-th iterate for a particular realization of our algorithm.

We also list here, for convenience, several constants that are used in this chapter to bound various quantities. These constants are denoted by κ with subscripts indicating quantities that they are meant to bound.

κ_{ef} "error in the function value",

κ_{eg} "error in the gradient",

κ_{Eef} "expectation of the error in the function value",

κ_{fcd} "fraction of Cauchy decrease",

κ_{bhm} "bound on the Hessian of the models",

κ_{et} "error in Taylor expansion".

2.3 Trust Region Method

We consider the trust-region class of methods for minimization of unconstrained functions. They operate as follows: at each iteration k, given the current iterate x_k and a trust-region radius δ_k, a (random) model $m_k(x)$ is built, which serves as an approximation of $f(x)$ in $B(x_k, \delta_k)$. The model is assumed to be of the form

$$m_k(x_k + s) = f_k + g_k^\top s + s^\top H_k s. \tag{2.2}$$

It is possible to generalize our framework to other forms of models, as long as all conditions on the models, described below, hold. We consider quadratic models for simplicity of the

presentation and because they are the most common. The model $m_k(x)$ is (approximately) minimized in $B(x_k, \delta_k)$ to produce a step s_k. (Random) estimates of $f(x_k)$ and $f(x_k + s_k)$ are obtained, denoted by f_k^0 and f_k^s respectively. The achieved reduction is measured by comparing f_k^0 and f_k^s; if reduction is deemed sufficient, then $x_k + s_k$ is accepted as the next iterate x_{k+1}. Otherwise, the iterate remains x_k. The trust-region radius δ_{k+1} is then updated by either increasing or decreasing δ_k according to the outcome of the iteration. The details of the algorithm are presented in Algorithm 1.

Algorithm 1: STOCHASTIC DFO WITH RANDOM MODELS

1 **(Initialization):** Choose an initial point x_0 and an initial trust-region radius $\delta_0 \in (0, \delta_{\max})$ with $\delta_{\max} > 0$. Choose constants $\gamma > 1$, $\eta_1 \in (0, 1)$, $\eta_2 > 0$, set $k \leftarrow 0$.

2 **(Model construction):** Build a model $m_k(x_k + s) = f_k + g_k^\top s + s^\top H_k s$ that approximates $f(x)$ on $B(x_k, \delta_k)$ with $s = x - x_k$.

3 **(Step calculation):** Compute $s_k = \arg\min_{s:\|s\| \le \delta_k} m_k(s)$ (approximately) so that it satisfies condition (2.3).

4 **(Estimates calculation):** Obtain estimates f_k^0 and f_k^s of $f(x_k)$ and $f(x_k + s_k)$, respectively.

5 **(Acceptance of the trial point):** Compute $\rho_k = \dfrac{f_k^0 - f_k^s}{m_k(x_k) - m_k(x_k + s_k)}$. If $\rho_k \ge \eta_1$ and $\|g_k\| \ge \eta_2 \delta_k$, set $x_{k+1} = x_k + s_k$; otherwise, set $x_{k+1} = x_k$.

6 **(Trust-region radius update):** If $\rho_k \ge \eta_1$ and $\|g_k\| \ge \eta_2 \delta_k$, set $\delta_{k+1} = \min\{\gamma \delta_k, \delta_{\max}\}$; otherwise $\delta_{k+1} = \gamma^{-1} \delta_k$; $k \leftarrow k + 1$ and go to step 1.

The trial step computed on each iteration has to provide sufficient decrease of the model; in other words it has to satisfy the following standard fraction of Cauchy decrease condition:

Assumption 2.3.1. *For every k, the step s_k is computed so that*

$$m_k(x_k) - m_k(x_k + s_k) \ge \frac{\kappa_{fcd}}{2} \|g_k\| \min\left\{ \frac{\|g_k\|}{\|H_k\|}, \delta_k \right\} \tag{2.3}$$

for some constant $\kappa_{fcd} \in (0, 1]$.

If progress is achieved and a new iterate is accepted in the k-th iteration, then we refer to the iteration as a *successful iteration*. Otherwise, the iteration is unsuccessful (and no step is taken). Hence a successful iteration occurs when $\rho_k \ge \eta_1$ and $\|g_k\| \ge \eta_2 \delta_k$. However, a successful iteration does not necessarily yield an actual reduction in the true function

f. This is because the values of $f(x)$ are not accessible in our stochastic setting and the step acceptance decision is made merely based on the estimates of $f(x_k)$ and $f(x_k + s_k)$. If these estimates, f_k^0 and f_k^s, are not accurate enough, a successful iteration can result in an increase of the true function value. Hence we consider two types of successful iterations - those where $f(x)$ is in fact decreased proportionally to $f_k^0 - f_k^s$ which we call *true* successful iterations, and all other successful iterations, where the decrease of $f(x)$ can be arbitrarily small or even negative, which we call *false* successful iterations. Our setting and algorithmic framework do not allow us to determine which successful iterations are true and which ones are false; however, we will be able to show that true successful iterations occur sufficiently often for convergence to hold if the random estimates f_k^0 and f_k^s are sufficiently accurate.

A trust region framework based on random models was introduced and analyzed in [4]. In that paper, the authors introduced the concept of probabilistically fully-linear models to determine the conditions that random models should satisfy for convergence of the algorithm to hold. However, the randomness in the models in their setting arises from the construction process, and not from the noisy objective function. It is assumed in [4] that the function values at the current iterate and the trial point can be computed exactly and hence all successful iterations are true in that case. In our case, it is necessary to define a measure for the accuracy of the estimates f_k^0 and f_k^s (which, as we will see, generally has to be tighter than the measure of accuracy of the model). We will use a modified version of the probabilistic estimates introduced in [48].

2.4 Probabilistic Models and Estimates

The models in this chapter are functions which are constructed on each iteration, based on random samples of the stochastic function $\tilde{f}(x)$ (and possibly its derivatives). Hence, the models themselves are random and so is their behavior and influence on the iterations. Hence, M_k will denote a random model in the k-th iteration, while we will use the notation $m_k = M_k(\omega)$ for its realizations. As a consequence of using random models, the iterates X_k, the trust-region radii Δ_k and the steps S_k are also random quantities, and so $x_k = X_k(\omega)$, $\delta_k = \Delta_k(\omega)$, $s_k = S_k(\omega)$ will denote their respective realizations. Similarly, let random

quantities $\{F_k^0, F_k^s\}$ denote the estimates of $f(X_k)$ and $f(X_k + S_k)$, with their realizations denoted by $f_k^0 = F_k^0(\omega)$ and $f_k^s = F_k^s(\omega)$. In other words, Algorithm 1 results in a stochastic process $\{M_k, X_k, S_k, \Delta_k, F_k^0, F_k^s\}$. Our goal is to show that, under certain conditions on the sequences $\{M_k\}$ and $\{F_k^0, F_k^s\}$, the resulting stochastic process has desirable convergence properties with probability one. In particular, we will assume that models M_k and estimates F_k^0, F_k^s are sufficiently accurate with sufficiently high probability, conditioned on the past.

To formalize conditioning on the past, let $\mathcal{F}_{k-1}^{M \cdot F}$ denote the σ-algebra generated by $M_0, \cdots, M_{k-1}$ and $F_0, \cdots, F_{k-1}$ and let $\mathcal{F}_{k-1/2}^{M \cdot F}$ denote the σ-algebra generated by $M_0, \cdots, M_k$ and $F_0, \cdots, F_{k-1}$.

To formalize sufficient accuracy, let us recall a measure for the accuracy of deterministic models introduced in [20] and [21] (with the exact notation introduced in [8]).

Definition 2.4.1. *Suppose ∇f is Lipschitz continuous. A function m_k is a* κ-fully linear model of f on $B(x_k, \delta_k)$ *provided, for $\kappa = (\kappa_{ef}, \kappa_{eg})$ and $\forall y \in B$,*

$$\begin{aligned} \|\nabla f(y) - \nabla m_k(y)\| &\leq \kappa_{eg}\delta_k, \text{ and} \\ |f(y) - m_k(y)| &\leq \kappa_{ef}\delta_k^2. \end{aligned} \tag{2.4}$$

In this chapter, we extend the following concept of probabilistically fully-linear models which was proposed in [4].

Definition 2.4.2. *A sequence of random models $\{M_k\}$ is said to be* α-probabilistically κ-fully linear *with respect to the corresponding sequence $\{B(X_k, \Delta_k)\}$ if the events*

$$I_k = \{M_k \text{ is a } \kappa\text{-fully linear model of } f \text{ on } B(X_k, \Delta_k)\} \tag{2.5}$$

satisfy the condition

$$P(I_k | \mathcal{F}_{k-1}^M) \geq \alpha,$$

where $\mathcal{F}_{k-1}^M$ is the σ-algebra generated by $M_0, \cdots, M_{k-1}$.

These probabilistically fully-linear models have the very simple properties that they are fully-linear (i.e., accurate enough) with sufficiently high probability conditioned on

the past, and they can be arbitrarily inaccurate otherwise. This property is somewhat different from the properties of models typical to stochastic optimization (such as, for example, stochastic gradient-based models), where assumptions on the expected value and the variance of the models is imposed. We will discuss this in more detail in Section 2.6.

In this chapter, aside from sufficiently accurate models, we require estimates of the function values $f(x_k)$, $f(x_k + s_k)$ that are sufficiently accurate. This is needed in order to evaluate whether a step is successful, unlike the case in [4] where the exact values $f(x_k)$ and $f(x_k + s_k)$ are assumed to be available. The following definition of accurate estimates is a modified version of that used in [48].

Definition 2.4.3. *The estimates f_k^0 and f_k^s are said to be* ϵ_F-accurate estimates *of $f(x_k)$ and $f(x_k + s_k)$, respectively, for a given δ_k if*

$$|f_k^0 - f(x_k)| \leq \epsilon_F \delta_k^2 \text{ and } |f_k^s - f(x_k + s_k)| \leq \epsilon_F \delta_k^2. \tag{2.6}$$

We now modify Definitions 2.4.2 and 2.4.3 and introduce definitions of probabilistically accurate models and estimates which we will use throughout the remainder of the chapter.

Definition 2.4.4. *A sequence of random models $\{M_k\}$ is said to be* α-probabilistically κ-fully linear *with respect to the corresponding sequence $\{B(X_k, \Delta_k)\}$ if the events*

$$I_k = \{M_k \text{ is a } \kappa\text{-fully linear model of } f \text{ on } B(X_k, \Delta_k)\} \tag{2.7}$$

satisfy the condition

$$P(I_k | \mathcal{F}_{k-1}^{M \cdot F}) \geq \alpha,$$

where $\mathcal{F}_{k-1}^{M \cdot F}$ is the σ-algebra generated by $M_0, \cdots, M_{k-1}$ and $F_0, \cdots, F_{k-1}$.

Definition 2.4.5. *A sequence of random estimates $\{F_k^0, F_k^s\}$ is said to be* β-probabilistically ϵ_F-accurate *with respect to the corresponding sequence $\{X_k, \Delta_k, S_k\}$ if the events*

$$J_k = \{F_k^0, F_k^s \text{ are } \epsilon_F\text{-accurate estimates of } f(x_k) \text{ and } f(x_k + s_k), \text{ respectively, for } \Delta_k\} \tag{2.8}$$

satisfy the condition

$$P(J_k|\mathcal{F}_{k-1/2}^{M\cdot F}) \geq \beta,$$

where ϵ_F *is a fixed constant and* $\mathcal{F}_{k-1/2}^{M\cdot F}$ *is the* σ*-algebra generated by* $M_0, \cdots, M_k$ *and* $F_0, \cdots, F_{k-1}$.

Motivated by Definitions 2.4.4 and 2.4.5, we will make later an assumption in our analysis that our method has access to a sequence of α-probabilistically κ-fully linear models, for some fixed $\kappa = (\kappa_{ef}, \kappa_{eg})$ and to a sequence of β-probabilistically ϵ_F-accurate estimates, for some fixed, sufficiently small ϵ_F. This will imply that the model and the estimate accuracy are both assumed to be proportional to δ_k^2 (with some probability); we remark now that the condition on the estimates will be somewhat tighter due to an upper bound on ϵ_F. However, we will see that this upper bound is not too small.

Procedures for obtaining probabilistically fully-linear models and probabilistically accurate estimates under different models of noise are discussed in Section 2.6.

2.5 Convergence Analysis

We now present first-order convergence analysis for the general framework described in Algorithm 1. Towards that end, we assume that the function f and its gradient are Lipschitz continuous in regions considered by the algorithm realizations.

Assumption 2.5.1 (Assumptions on f)**.** *Let* x_0 *and* $\delta_{\max}$ *be given. Let* $\mathcal{L}(x_0)$ *define the set in* $\mathbb{R}^n$ *which contains all iterates of our algorithm. Assume that* f *is bounded from below on* $\mathcal{L}(x_0)$*. Assume also that the function* f *and its gradient* ∇f *are* L*-Lipschitz continuous on the set* $\mathcal{L}_{enl}(x_0)$*, where* $\mathcal{L}_{enl}(x_0)$ *defines the region considered by the algorithm realizations*

$$\mathcal{L}_{enl}(x_0) = \bigcup_{x \in \mathcal{L}(x_0)} B(x; \delta_{\max}).$$

Remark 2.5.1. *In the case of deterministic functions* $\mathcal{L}(x_0) = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$*, because algorithm iterates never increase the objective function value, hence they do not step*

outside the initial level set. However, here we allow iterates to increase the function value, because the true function value is not known. Such iterates, as we will see, may happen with some (relatively small) probability, hence the algorithm can venture outside the initial level set. Hence we choose to make the assumption above, which of course depends on the algorithmic behavior. Clearly, if we assume a global Lipschitz constant and global lower bound, then the above assumption always holds. If we prefer to weaken this assumption, then there are several algorithmic remedies possible; however, they will make our analysis more complicated and we choose to leave it for future work.

The second assumption provides a uniform upper bound on the model Hessian.

Assumption 2.5.2. *There exists a positive constant κ_{bhm} such that, for every k, the Hessian H_k of all realizations m_k of M_k satisfy*

$$\|H_k\| \leq \kappa_{bhm}.$$

Note that since we are concerned with convergence to a first-order stationary point in this chapter, the bound κ_{bhm} can be chosen to be any nonnegative number, including zero. Allowing a larger bound will give more flexibility to the algorithm and may allow better Hessian approximations, but as we will see in the convergence analysis, this imposes restrictions on the trust region radius and some other algorithmic parameters.

We now state the following result from martingale literature (see [27], Exercise 5.3.1) that will be useful later in our analysis.

Theorem 2.5.1. *Let G_k be a submartingale, i.e., a sequence of random variables which, for every k,*

$$E[G_k|\mathcal{F}_{k-1}^G] \geq G_{k-1},$$

where $\mathcal{F}_{k-1}^G = \sigma(G_0, \ldots, G_{k-1})$ is the σ-algebra generated by $G_0, \ldots, G_{k-1}$, and $E[G_k|\mathcal{F}_{k-1}^G]$ denotes the conditional expectation of G_k given the past history of events $\mathcal{F}_{k-1}^G$.

Assume further that $G_k - G_{k-1} \leq M < \infty$, for every k. Then,

$$P\left(\left\{\lim_{k\to\infty} G_k < \infty\right\} \cup \left\{\limsup_{k\to\infty} G_k = \infty\right\}\right) = 1. \tag{2.9}$$

We now prove some auxiliary lemmas that provide conditions under which decrease of the true objective function $f(x)$ is guaranteed. The first lemma states that if the trust-region radius is small enough relative to the size of the model gradient and if the model is fully linear, then the step s_k provides a decrease in $f(x)$ proportional to the size of the model gradient. Note that the trial step may still be rejected if the estimates f_k^0 and f_k^s are not accurate enough.

Lemma 2.5.1. *Suppose that a model m_k of the form (2.2) is a $(\kappa_{ef}, \kappa_{eg})$-fully linear model of f on $B(x_k, \delta_k)$. If*

$$\delta_k \leq \min\left\{\frac{1}{\kappa_{bhm}}, \frac{\kappa_{fcd}}{8\kappa_{ef}}\right\}\|g_k\|,$$

then the trial step s_k leads to an improvement in $f(x_k + s_k)$ such that

$$f(x_k + s_k) - f(x_k) \leq -\frac{\kappa_{fcd}}{4}\|g_k\|\delta_k. \tag{2.10}$$

Proof. Using the Cauchy decrease condition, the upper bound on the model Hessian, and the fact that $\|g_k\| \geq \kappa_{bhm}\delta_k$, we have

$$m_k(x_k) - m_k(x_k + s_k) \geq \frac{\kappa_{fcd}}{2}\|g_k\| \min\left\{\frac{\|g_k\|}{\|H_k\|}, \delta_k\right\} = \frac{\kappa_{fcd}}{2}\|g_k\|\delta_k.$$

Since the model is κ-fully linear, one can express the improvement in f achieved by s_k as

$$\begin{aligned}
& f(x_k + s_k) - f(x_k) \\
= \;& f(x_k + s_k) - m(x_k + s_k) + m(x_k + s_k) - m(x_k) + m(x_k) - f(x_k) \\
\leq \;& 2\kappa_{ef}\delta_k^2 - \frac{\kappa_{fcd}}{2}\|g_k\|\delta_k \\
\leq \;& -\frac{\kappa_{fcd}}{4}\|g_k\|\delta_k,
\end{aligned}$$

where the last inequality is implied by $\delta_k \leq \frac{\kappa_{fcd}}{8\kappa_{ef}}\|g_k\|$. □

The next lemma shows that for δ_k small enough relative to the size of the true gradient $\nabla f(x_k)$, the guaranteed decrease in the objective function provided by s_k is proportional

to the size of the true gradient.

Lemma 2.5.2. *Under Assumption 2.5.2, suppose that a model is $(\kappa_{ef}, \kappa_{eg})$-fully linear on $B(x_k, \delta_k)$. If*

$$\delta_k \leq \min \left\{ \frac{1}{\kappa_{bhm} + \kappa_{eg}}, \frac{1}{\frac{8\kappa_{ef}}{\kappa_{fcd}} + \kappa_{eg}} \right\} \|\nabla f(x_k)\|, \tag{2.11}$$

then the trial step s_k leads to an improvement in $f(x_k + s_k)$ such that

$$f(x_k + s_k) - f(x_k) \leq -C_1 \|\nabla f(x_k)\| \delta_k, \tag{2.12}$$

where $C_1 = \frac{\kappa_{fcd}}{4} \cdot \max \left\{ \frac{\kappa_{bhm}}{\kappa_{bhm}+\kappa_{eg}}, \frac{8\kappa_{ef}}{8\kappa_{ef}+\kappa_{fcd}\kappa_{eg}} \right\}$.

Proof. The definition of a κ-fully-linear model yields that

$$\|g_k\| \geq \|\nabla f(x)\| - \kappa_{eg}\delta_k.$$

Since condition (2.11) implies that $\|\nabla f(x_k)\| \geq \max \left\{ \kappa_{bhm} + \kappa_{eg}, \frac{8\kappa_{ef}}{\kappa_{fcd}} + \kappa_{eg} \right\} \delta_k$, we have

$$\|g_k\| \geq \max \left\{ \kappa_{bhm}, \frac{8\kappa_{ef}}{\kappa_{fcd}} \right\} \delta_k.$$

Hence, the conditions of Lemma 2.5.1 hold and we have

$$f(x_k + s_k) - f(x_k) \leq -\frac{\kappa_{fcd}}{4} \|g_k\| \delta_k. \tag{2.13}$$

Since $\|g_k\| \geq \|\nabla f(x)\| - \kappa_{eg}\delta_k$ and δ_k satisfies (2.11), we also have

$$\|g_k\| \geq \max \left\{ \frac{\kappa_{bhm}}{\kappa_{bhm} + \kappa_{eg}}, \frac{8\kappa_{ef}}{8\kappa_{ef} + \kappa_{fcd}\kappa_{eg}} \right\} \|\nabla f(x_k)\|. \tag{2.14}$$

Combining (2.13) and (2.14) yields (2.12). □

We now prove a lemma that states that if a) the estimates are sufficiently accurate, b) the model is fully-linear, and c) the trust-region radius is sufficiently small relative to the

size of the model gradient, then a successful step is guaranteed.

Lemma 2.5.3. *Under Assumption 2.5.2, suppose that m_k is $(\kappa_{ef}, \kappa_{eg})$-fully linear on $B(x_k, \delta_k)$ and the estimates $\{f_k^0, f_k^s\}$ are ϵ_F-accurate with $\epsilon_F \leq \kappa_{ef}$. If*

$$\delta_k \leq \min\left\{\frac{1}{\kappa_{bhm}}, \frac{1}{\eta_2}, \frac{\kappa_{fcd}(1-\eta_1)}{8\kappa_{ef}}\right\}\|g_k\|, \tag{2.15}$$

then the k-th iteration is successful.

Proof. Since $\delta_k \leq \frac{\|g_k\|}{\kappa_{bhm}}$, the Cauchy decrease condition and the uniform bound on H_k immediately yield that

$$m_k(x_k) - m_k(x_k + s_k) \geq \frac{\kappa_{fcd}}{2}\|g_k\| \min\left\{\frac{\|g_k\|}{\kappa_{bhm}}, \delta_k\right\} = \frac{\kappa_{fcd}}{2}\|g_k\|\delta_k. \tag{2.16}$$

The model m_k being $(\kappa_{ef}, \kappa_{eg})$-fully linear implies that

$$|f(x_k) - m_k(x_k)| \leq \kappa_{ef}\delta_k^2, \text{ and} \tag{2.17}$$

$$|f(x_k + s_k) - m_k(x_k + s_k)| \leq \kappa_{ef}\delta_k^2. \tag{2.18}$$

Since the estimates are ϵ_F-accurate with $\epsilon_F \leq \kappa_{ef}$, we obtain

$$|f_k^0 - f(x_k)| \leq \kappa_{ef}\delta_k^2, \text{ and } |f_k^s - f(x_k + s_k)| \leq \kappa_{ef}\delta_k^2. \tag{2.19}$$

We have

$$\begin{aligned}
\rho_k &= \frac{f_k^0 - f_k^s}{m_k(x_k) - m_k(x_k + s_k)} \\
&= \frac{f_k^0 - f(x_k)}{m_k(x_k) - m_k(x_k + s_k)} + \frac{f(x_k) - m_k(x_k)}{m_k(x_k) - m_k(x_k + s_k)} + \frac{m_k(x_k) - m_k(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)} \\
&\quad + \frac{m_k(x_k + s_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)} + \frac{f(x_k + s_k) - f_k^s}{m_k(x_k) - m_k(x_k + s_k)},
\end{aligned}$$

which, combined with (2.16)-(2.19), implies

$$|\rho_k - 1| \leq \frac{8\kappa_{ef}\delta_k^2}{\kappa_{fcd}\|g_k\|\delta_k} \leq 1 - \eta_1,$$

where we have used the assumption $\delta_k \leq \frac{\kappa_{fcd}(1-\eta_1)}{8\kappa_{ef}}\|g_k\|$ to deduce the last inequality. Hence, $\rho_k \geq \eta_1$. Moreover, since $\|g_k\| \geq \eta_2\delta_k$, the k-th iteration is successful. □

Finally, we state and prove a lemma which guarantees an amount of decrease of the objective function on a true successful iteration.

Lemma 2.5.4. *Under Assumption 2.5.2, suppose that the estimates $\{f_k^0, f_k^s\}$ are ϵ_F-accurate with $\epsilon_F < \frac{1}{4}\eta_1\eta_2\kappa_{fcd}\min\left\{\frac{\eta_2}{\kappa_{bhm}}, 1\right\}$. If a trial step s_k is accepted (a successful iteration occurs), then the improvement in f is bounded below as follows*

$$f(x_{k+1}) - f(x_k) \leq -C_2\delta_k^2, \tag{2.20}$$

where $C_2 = \frac{1}{2}\eta_1\eta_2\kappa_{fcd}\min\left\{\frac{\eta_2}{\kappa_{bhm}}, 1\right\} - 2\epsilon_F > 0$.

Proof. An iteration being successful indicates that $\|g_k\| \geq \eta_2\delta_k$ and $\rho \geq \eta_1$. Thus,

$$\begin{aligned} f_k^0 - f_k^s &\geq \eta_1(m_k(x_k) - m_k(x_k + s_k)) \\ &\geq \eta_1\frac{\kappa_{fcd}}{2}\|g_k\|\min\left\{\frac{\|g_k\|}{\|H_k\|}, \delta_k\right\} \\ &\geq \frac{1}{2}\eta_1\eta_2\kappa_{fcd}\min\left\{\frac{\eta_2}{\kappa_{bhm}}, 1\right\}\delta_k^2. \end{aligned}$$

Then, since the estimates are ϵ_F-accurate, we have that the improvement in f can be bounded as

$$f(x_k + s_k) - f(x_k) = f(x_k + s_k) - f_k^s + f_k^s - f_k^0 + f_k^0 - f(x_k) \leq -C_2\delta_k^2,$$

where $C_2 = \frac{1}{2}\eta_1\eta_2\kappa_{fcd}\min\left\{\frac{\eta_2}{\kappa_{bhm}}, 1\right\} - 2\epsilon_F > 0$. □

To prove convergence of Algorithm 1, we will need to assume that the models $\{M_k\}$ and estimates $\{F_k^0, F_k^s\}$ are sufficiently accurate with sufficiently high probability.

Assumption 2.5.3. *Given values of $\alpha, \beta \in (0, 1)$ and $\epsilon_F > 0$, there exist κ_{eg} and κ_{ef} such that the sequence of models $\{M_k\}$ and estimates $\{F_k^0, F_k^s\}$ generated by Algorithm 1 are, respectively, α-probabilistically $(\kappa_{ef}, \kappa_{eg})$- fully-linear and β-probabilistically ϵ_F-accurate.*

Remark 2.5.2. *Note that this assumption is a statement about the existence of constants $\kappa = (\kappa_{ef}, \kappa_{eg})$ given an α, β and ϵ_F - we will determine exact conditions on α, β and ϵ_F in Theorem 2.5.2 and Lemma 2.5.1 below.*

The following theorem states that the trust-region radius converges to zero with probability 1.

Theorem 2.5.2. *Let Assumptions 2.5.1 and 2.5.2 be satisfied and assume that in Algorithm 1 the following holds.*

- *The step acceptance parameter η_2 is chosen so that*

$$\eta_2 \geq \max\left\{\kappa_{bhm}, \frac{8\kappa_{ef}}{\kappa_{fcd}(1-\eta_1)}\right\}. \tag{2.21}$$

- *The accuracy parameter of the estimates satisfies*

$$\epsilon_F \leq \min\left\{\kappa_{ef}, \frac{1}{8}\eta_1\eta_2\kappa_{fcd}\right\}. \tag{2.22}$$

Then α and β can be chosen so that, if Assumption 2.5.3 holds for these values, then the sequence of trust-region radii generated by Algorithm 1, $\{\Delta_k\}$, satisfies

$$\sum_{k=0}^{\infty} \Delta_k^2 < \infty \tag{2.23}$$

almost surely.

Proof. We base our proof on properties of the random function $\Phi_k = \nu f(X_k) + (1-\nu)\Delta_k^2$, where $\nu \in (0,1)$ is a fixed constant, which is specified below. A similar function is used in the analysis in [48], but the analysis itself is different. The overall goal is to show that there exists a constant $\sigma > 0$ such that for all k

$$E[\Phi_{k+1} - \Phi_k | \mathcal{F}_{k-1}^{M \cdot F}] \leq -\sigma\Delta_k^2 < 0. \tag{2.24}$$

Since f is bounded from below and $\Delta_k > 0$, we have that Φ_k is bounded from below for all k; hence if (2.24) holds on every iteration, then by summing (2.24) over $k \in (1, \infty)$ and

taking expectations on both sides we can conclude that (2.23) holds with probability 1. Hence, to prove the theorem we need to show that (2.24) holds on each iteration.

Let us pick some constant ζ which satisfies

$$\zeta \geq \kappa_{eg} + \max\left\{\eta_2, \frac{8\kappa_{ef}}{\kappa_{fcd}(1-\eta_1)}\right\}. \tag{2.25}$$

We now consider two possible cases: $\|\nabla f(x_k)\| \geq \zeta\delta_k$ and $\|\nabla f(x_k)\| < \zeta\delta_k$. We will show that (2.24) holds in both cases and hence it holds on every iteration. Given ζ we now select $\nu \in (0,1)$ such that

$$\frac{\nu}{1-\nu} \quad > \quad \max\left\{\frac{4\gamma^2}{\zeta C_1}, \frac{4\gamma^2}{\eta_1\eta_2\kappa_{fcd}}, \frac{\gamma^2}{\kappa_{ef}}\right\}, \tag{2.26}$$

with C_1 defined as in Lemma 2.5.2.

As usual, let x_k, δ_k, s_k, g_k, and ϕ_k denote realizations of random quantities X_k, Δ_k, S_k, G_k, and Φ_k, respectively.

Let us consider any realization of Algorithm 1. Note that on all successful iterations, $x_{k+1} = x_k + s_k$ and $\delta_{k+1} = \min\{\gamma\delta_k, \delta_{max}\}$ with $\gamma > 1$, hence

$$\phi_{k+1} - \phi_k \leq \nu(f(x_{k+1}) - f(x_k)) + (1-\nu)(\gamma^2 - 1)\delta_k^2. \tag{2.27}$$

On all unsuccessful iterations, $x_{k+1} = x_k$ and $\delta_{k+1} = \frac{1}{\gamma}\delta_k$, i.e.

$$\phi_{k+1} - \phi_k = (1-\nu)(\frac{1}{\gamma^2} - 1)\delta_k^2 \equiv b_1 < 0. \tag{2.28}$$

For each iteration and each of the two cases we consider, we will analyze the four possible combined outcomes of the events I_k and J_k as defined in (2.7) and (2.8), respectively.

Before presenting the formal proof let us outline the key ideas. We will show that, unless both the model and the estimates are bad on iteration k, our choice of $\nu \in (0,1)$ being sufficiently close to 1 causes the decrease in ϕ_k on a successful iteration (2.27) to be greater than the decrease in ϕ_k on an unsuccessful iteration (which is equal to b_1, according to (2.28)). When the model and the estimates are both bad, an increase in ϕ_k

may occur. This increase is bounded by a value proportional to δ_k^2 when $\|\nabla f(x_k)\| < \zeta\delta_k$. When $\|\nabla f(x_k)\| \geq \zeta\delta_k$, though, the increase in ϕ_k may be proportional to $\|\nabla f(x_k)\|\delta_k$. However, since iterations with good models and good estimates will provide *decrease in* ϕ_k *also proportional to* $\|\nabla f(x_k)\|\delta_k$, then by choosing values of α and β close enough to 1, we can ensure that ϕ_k decreases *in expectation.*

We now present the proof.

Case 1: $\|\nabla f(x_k)\| \geq \zeta\delta_k$**.**

a. I_k and J_k are both true, i.e., both the model and the estimates are good on iteration k. From the definition of ζ, we know

$$\|\nabla f(x_k)\| \geq \left(\kappa_{eg} + \max\left\{\eta_2, \frac{8\kappa_{ef}}{\kappa_{fcd}(1-\eta_1)}\right\}\right)\delta_k.$$

Then since the model m_k is κ-fully linear and, from $\eta_2 > \kappa_{bhm}$, $\epsilon_F \leq \kappa_{ef}$ and $0 < \eta_1 < 1$, it is easy to show that the condition (2.11) in Lemma 2.5.2 holds. Therefore, the trial step s_k leads to a decrease in f as in (2.12).

Moreover, since

$$\|g_k\| \geq \|\nabla f(x_k)\| - \kappa_{eg}\delta_k \geq (\zeta - \kappa_{eg})\delta_k \geq \max\left\{\eta_2, \frac{8\kappa_{ef}}{\kappa_{fcd}(1-\eta_1)}\right\}\delta_k,$$

and the estimates $\{f_k^0, f_k^s\}$ are ϵ_F-accurate, with $\epsilon_F \leq \kappa_{ef}$, the condition (2.15) in Lemma 2.5.3 holds. Hence, iteration k is successful, i.e. $x_{k+1} = x_k + s_k$ and $\delta_{k+1} = \gamma\delta_k$.

Combining (2.12) and (2.27), we get

$$\phi_{k+1} - \phi_k \leq -\nu C_1\|\nabla f(x_k)\|\delta_k + (1-\nu)(\gamma^2 - 1)\delta_k^2 \equiv b_2, \tag{2.29}$$

with C_1 defined in Lemma 2.5.2. Since $\|\nabla f(x_k)\| \geq \zeta\delta_k$ we have

$$b_2 \leq [-\nu C_1\zeta + (1-\nu)(\gamma^2 - 1)]\delta_k^2 < 0, \tag{2.30}$$

for $\nu \in (0, 1)$ satisfying (2.26).

b. I_k is true and J_k is false, i.e., we have a good model and bad estimates on iteration k.

   In this case, Lemma 2.5.2 still holds, that is s_k yields a sufficient decrease in f; hence, if the iteration is successful, we obtain (2.29) and (2.30). However, the step can be erroneously rejected, because of inaccurate probabilistic estimates, in which case we have an unsuccessful iteration and (2.28) holds. Since (2.26) holds, the right hand side of the first relation in (2.30) is strictly smaller than the right hand side of the first relation in (2.28) and therefore, (2.28) holds whether the iteration is successful or not.

c. I_k is false and J_k is true, i.e., we have a bad model and good estimates on iteration k. In this case, iteration k can be either successful or unsuccessful. In the unsuccessful case, (2.28) holds. When the iteration is successful, since the estimates are ϵ_F-accurate and (2.22) holds, then by Lemma 2.5.4, (2.20) holds with $C_2 \geq \frac{1}{4}\eta_1\eta_2\kappa_{fcd}$. Hence, in this case, we have

$$\phi_{k+1} - \phi_k \leq [-\nu C_2 + (1 - \nu)(\gamma^2 - 1)]\delta_k^2. \tag{2.31}$$

   Again, due to the choice of ν satisfying (2.26) we have that, as in case (b), (2.28) holds whether the iteration is successful or not.

d. I_k and J_k are both false, i.e., both the model and the estimates are bad on iteration k.

   Inaccurate estimates can cause the algorithm to accept a bad step, which may lead to an increase both in f and in δ_k. Hence in this case $\phi_{k+1} - \phi_k$ may be positive. However, combining the Taylor expansion of $f(x_k)$ at $x_k + s_k$ and the Lipschitz continuity of $\nabla f(x)$ we can bound the amount of increase in f, hence bounding $\phi_{k+1} - \phi_k$ from above. By adjusting the probability of outcome (d) to be sufficiently small, we can ensure that in expectation Φ_k is sufficiently reduced.

In particular, from Taylor's Theorem and the L-Lipschitz continuity of $\nabla f(x)$ we have, respectively,

$$\begin{aligned} f(x_k) - f(x_k + s_k) &\geq \nabla f(x_k + s_k)^\top (-s_k) - \frac{1}{2} L \delta_k^2, \text{ and} \\ \|\nabla f(x_k + s_k) - \nabla f(x_k)\| &\leq L s_k \leq L \delta_k. \end{aligned}$$

From this we can derive that any increase of $f(x_k)$ is bounded by

$$f(x_k + s_k) - f(x_k) \leq C_3 \|\nabla f(x_k)\| \delta_k,$$

where $C_3 = 1 + \frac{3L}{2\zeta}$. Hence, the change in function ϕ is bounded:

$$\phi_{k+1} - \phi_k \leq \nu C_3 \|\nabla f(x_k)\| \delta_k + (1 - \nu)(\gamma^2 - 1)\delta_k^2 \equiv b_3. \tag{2.32}$$

Now we are ready to take the expectation of $\Phi_{k+1} - \Phi_k$ for the case when $\|\nabla f(X_k)\| \geq \zeta \Delta_k$. We know that case (a) occurs with a probability at least $\alpha\beta$ (conditioned on the past) and in that case $\phi_{k+1} - \phi_k = b_2 < 0$ with b_2 defined in (2.29). Case (d) occurs with probability at most $(1 - \alpha)(1 - \beta)$ and in that case $\phi_{k+1} - \phi_k$ is bounded from above by $b_3 > 0$. Cases (b) and (c) occur otherwise and in those cases $\phi_{k+1} - \phi_k$ is bounded from above by $b_1 < 0$, with b_1 defined in (2.28). Finally we note that $b_1 > b_2$ due to our choice of ν in (2.26).

Hence, we can combine (2.28), (2.29), (2.31), and (2.32), and use B_1, B_2, and B_3 as random counterparts of b_1, b_2, and b_3, to obtain the following bound

$$\begin{aligned} & E[\Phi_{k+1} - \Phi_k | \mathcal{F}_{k-1}^{M \cdot F}, \{\|\nabla f(X_k)\| \geq \zeta \Delta_k\}] \\ \leq\ & \alpha\beta B_2 + [\alpha(1 - \beta) + (1 - \alpha)\beta] B_1 + (1 - \alpha)(1 - \beta) B_3 \\ =\ & \alpha\beta[-\nu C_1 \|\nabla f(X_k)\| \Delta_k + (1 - \nu)(\gamma^2 - 1)\Delta_k^2] \\ & + [\alpha(1 - \beta) + (1 - \alpha)\beta](1 - \nu)(\frac{1}{\gamma^2} - 1)\Delta_k^2 \\ & + (1 - \alpha)(1 - \beta)[\nu C_3 \|\nabla f(X_k)\| \Delta_k + (1 - \nu)(\gamma^2 - 1)\Delta_k^2]. \end{aligned}$$

Rearranging the terms we obtain

$$\begin{aligned} & E[\Phi_{k+1} - \Phi_k | \mathcal{F}_{k-1}^{M \cdot F}, \{\|\nabla f(X_k)\| \geq \zeta \Delta_k\}] \\ \leq \; & [-\nu C_1 \alpha\beta + (1-\alpha)(1-\beta)\nu C_3]\|\nabla f(X_k)\|\Delta_k \\ & + [\alpha\beta - \frac{1}{\gamma^2}(\alpha(1-\beta) + (1-\alpha)\beta) + (1-\alpha)(1-\beta)](1-\nu)(\gamma^2-1)\Delta_k^2 \\ \leq \; & [-C_1\alpha\beta + (1-\alpha)(1-\beta)C_3]\nu\|\nabla f(X_k)\|\Delta_k + (1-\nu)(\gamma^2-1)\Delta_k^2, \end{aligned}$$

where the last inequality holds because $\alpha\beta - \frac{1}{\gamma^2}(\alpha(1-\beta) + (1-\alpha)\beta) + (1-\alpha)(1-\beta) \leq [\alpha + (1-\alpha)][(\beta + (1-\beta)] = 1$.

Let us choose $0 < \alpha \leq 1$ and $0 < \beta \leq 1$ so that they satisfy

$$\frac{(\alpha\beta - \frac{1}{2})}{(1-\alpha)(1-\beta)} \;\; \geq \;\; \frac{C_3}{C_1} \tag{2.33}$$

which implies

$$[C_1\alpha\beta - (1-\alpha)(1-\beta)C_3] \geq \frac{1}{2}C_1 \geq 2\frac{(1-\nu)(\gamma^2-1)}{\nu\zeta},$$

where the last inequality is the result of (2.26). We note that the quantity $\frac{1}{2}$ in the numerator of (2.33) was chosen so that the first inequality of the above expression holds: see Remark 2.5.5 following the proof for a brief discussion about this choice.

Recall that $\|\nabla f(X_k)\| \geq \zeta\Delta_k$, hence

$$\begin{aligned} & [-C_1\alpha\beta + (1-\alpha)(1-\beta)C_3]\nu\|\nabla f(X_k)\|\Delta_k + (1-\nu)(\gamma^2-1)\Delta_k^2 \\ \leq \; & \frac{1}{2}[-C_1\alpha\beta + (1-\alpha)(1-\beta)C_3]\nu\|\nabla f(X_k)\|\Delta_k \leq -\frac{1}{4}C_1\nu\|\nabla f(X_k)\|\Delta_k. \end{aligned}$$

In summary, we have

$$E[\Phi_{k+1} - \Phi_k | \mathcal{F}_{k-1}^{M \cdot F}, \{\|\nabla f(X_k)\| \geq \zeta\Delta_k\}] \leq -\frac{1}{4}C_1\nu\|\nabla f(X_k)\|\Delta_k \tag{2.34}$$

and

$$E[\Phi_{k+1} - \Phi_k | \mathcal{F}_{k-1}^{M \cdot F}, \{\|\nabla f(X_k)\| \geq \zeta\Delta_k\}] \leq -\frac{1}{2}(1-\nu)(\gamma^2-1)\Delta_k^2. \tag{2.35}$$

For the purposes of this lemma and the liminf-type convergence result, which will follow, bound (2.35) is sufficient. We will use bound (2.34) in the proof of the lim-type convergence result.

Case 2: Let us consider now the iterations when $\|\nabla f(x_k)\| < \zeta\delta_k$. First we note that if $\|g_k\| < \eta_2\delta_k$, then we have an unsuccessful step and (2.28) holds. Hence, we now assume that $\|g_k\| \geq \eta_2\delta_k$ and again consider four possible outcomes. We will show that in all situations, except when both the model and the estimates are bad, (2.28) holds. In the remaining case, because $\|\nabla f(x_k)\| < \zeta\delta_k$, the increase in ϕ_k can be bounded from above by a multiple of δ_k^2. Hence by selecting appropriate values for probabilities α and β we will be able to establish the bound on expected decrease in Φ_k as in Case 1.

a. I_k and J_k are both true, i.e., both the model and the estimates are good on iteration k.

   The iteration may or may not be successful, even though I_k is true. On successful iterations, the good model ensures reduction in f. Applying the same argument as in the case 1(c) we establish (2.28).

b. I_k is true and J_k is false, i.e., we have a good model and bad estimates on iteration k.

   On unsuccessful iterations, (2.28) holds. On successful iterations, $\|g_k\| \geq \eta_2\delta_k$ and $\eta_2 \geq \kappa_{bhm}$ imply that

$$\begin{aligned} m_k(x_k) - m_k(x_k + s_k) &\geq \frac{\kappa_{fcd}}{2}\|g_k\| \min\left\{\frac{\|g_k\|}{\|H_k\|}, \delta_k\right\} \\ &\geq \eta_2\frac{\kappa_{fcd}}{2}\delta_k^2. \end{aligned}$$

   Since I_k is true, the model is κ-fully-linear, and the function decrease can be bounded

as

$$\begin{aligned} & f(x_k) - f(x_k + s_k) \\ = \; & f(x_k) - m_k(x_k) + m_k(x_k) - m_k(x_k + s_k) + m_k(x_k + s_k) - f(x_k + s_k) \\ \geq \; & (\eta_2 \frac{\kappa_{fcd}}{2} - 2\kappa_{ef})\delta_k^2 \geq \kappa_{ef}\delta_k^2 \end{aligned}$$

due to (2.21).

It follows that, if the k-th iterate is successful, then

$$\phi_{k+1} - \phi_k \leq [-\nu\kappa_{ef} + (1-\nu)(\gamma^2 - 1)]\delta_k^2. \tag{2.36}$$

Again by choosing $\nu \in (0, 1)$ so that (2.26) holds, we ensure that the right hand side of (2.36) is strictly smaller than that of (2.28), hence (2.28) holds, whether the iteration is successful or not.

Remark: η_2 may need to be a relatively large constant to satisfy (2.21). This is due to the fact that the model has to be sufficiently accurate to ensure decrease in the function if a step is taken, since the step is accepted based on poor estimates. Note that η_2 restricts the size of Δ_k, which is used both as a bound on the step size and the control of the accuracy. In general it is possible to have two separate quantities (related by a constant) - one to control the step size and another to control the accuracy. Hence, it is possible to modify our algorithm to accept steps larger than $\|g_k\|/\eta_2$. This will make the algorithm more practical, but the analysis much more complex. In this chapter, we choose to stay with the simplest version, but keep in mind that the condition (2.26) is not terminally restrictive.

c. I_k is false and J_k is true, i.e., we have a bad model and good estimates on iteration k.

   This case is analyzed identically to the case 1(c).

d. I_k and J_k are both false, i.e., both the model and the estimates are bad on iteration

k.

Here we bound the maximum possible increase in ϕ_k using the Taylor expansion and the Lipschitz continuity of $\nabla f(x)$.

$$f(x_k + s_k) - f(x_k) \leq \|\nabla f(x_k)\|\delta_k + \frac{1}{2}L\delta_k^2 < C_3\zeta\delta_k^2.$$

Hence, the change in function ϕ is

$$\phi_{k+1} - \phi_k \leq [\nu C_3\zeta + (1-\nu)(\gamma^2 - 1)]\delta_k^2. \tag{2.37}$$

We are now ready to bound the expectation of $\phi_{k+1} - \phi_k$ as we did in Case 1, except that in Case 2 we simply combine (2.37), which holds with probability at most $(1-\alpha)(1-\beta)$, and (2.28), which holds otherwise:

$$\begin{aligned}
& E[\Phi_{k+1} - \Phi_k | \mathcal{F}_{k-1}^{M\cdot F}, \{\|\nabla f(X_k)\| < \zeta\Delta_k\}] \\
\leq\ & [\alpha\beta + \alpha(1-\beta) + (1-\alpha)\beta](1-\nu)(\frac{1}{\gamma^2} - 1)\Delta_k^2 \\
& + (1-\alpha)(1-\beta)[\nu C_3\zeta + (1-\nu)(\gamma^2 - 1)]\Delta_k^2 \\
\leq\ & (1-\nu)(\frac{1}{\gamma^2} - 1)\Delta_k^2 + (1-\alpha)(1-\beta)[\nu C_3\zeta + (1-\nu)(\gamma^2 - \frac{1}{\gamma^2})]\Delta_k^2
\end{aligned}$$

If we choose probabilities $0 < \alpha \leq 1$ and $0 < \beta \leq 1$ so that the following holds,

$$(1-\alpha)(1-\beta) \leq \frac{\gamma^2 - 1}{\gamma^4 - 1 + 2\gamma^2 C_3\zeta \cdot \frac{\nu}{1-\nu}}, \tag{2.38}$$

then

$$E[\Phi_{k+1} - \Phi_k | \mathcal{F}_{k-1}^{M\cdot F}, \{\|\nabla f(X_k)\| < \zeta\Delta_k\}] \leq -\frac{1}{2}(1-\nu)(\frac{1}{1-\gamma^2})\Delta_k^2. \tag{2.39}$$

In conclusion, combining (2.35) and (2.39), and noting that $1 - \frac{1}{\gamma^2} < \gamma^2 - 1$ we have

$$E[\Phi_{k+1} - \Phi_k | \mathcal{F}_{k-1}^{M\cdot F}\}] \leq -\tfrac{1}{2}(1-\nu)(1 - \tfrac{1}{\gamma^2})\Delta_k^2 < 0,$$

which implies that (2.24) holds with $\sigma = -\frac{1}{2}(1-\nu)(1-\frac{1}{\gamma^2}-1) < 0$. This concludes the proof of the theorem. □

To summarize the conditions on the probabilities involved in Theorem 2.5.2 to ensure that the theorem holds, we state the following as a corollary.

Corollary 2.5.1. *Let all assumptions of Theorem 2.5.2 hold. The statement of Theorem 2.5.2 holds if the α and β are chosen to satisfy the following conditions:*

$$\frac{(\alpha\beta - \frac{1}{2})}{(1-\alpha)(1-\beta)} \geq \frac{1 + \frac{3L}{2\zeta}}{C_1} \tag{2.40}$$

and

$$(1-\alpha)(1-\beta) \leq \frac{\gamma^2 - 1}{\gamma^4 - 1 + \gamma^2 (3L + 2\zeta) \cdot \max\left\{\frac{4}{\zeta C_1}, \frac{4}{\eta_1 \eta_2 \kappa_{fcd}}, \frac{1}{\kappa_{ef}}\right\}}, \tag{2.41}$$

with $C_1 = \frac{\kappa_{fcd}}{4} \cdot \max\left\{\frac{\kappa_{bhm}}{\kappa_{bhm}+\kappa_{eg}}, \frac{8\kappa_{ef}}{8\kappa_{ef}+\kappa_{fcd}\kappa_{eg}}\right\}$ and $\zeta = \kappa_{eg} + \eta_2$.

Proof. The proof follows simply from combining the expression for C_3 and condition (2.26) with (2.33) and (2.38). □

Clearly, choosing α and β sufficiently close to 1 will satisfy the conditions given in (2.40) and (2.41).

Remark 2.5.3. *We will briefly illustrate through a simple example how these algorithmic parameters scale with problem data.*

Recall that L is the Lipschitz constant of ∇f and of f over $\mathcal{L}_{enl}(x^0)$. It is reasonable to expect that κ_{ef} and κ_{eg} are quantities that scale with L, since Taylor models satisfy this condition, as do polynomial interpolation and regression models based on well-poised data sets [21]. Let us assume for the sake of an example that $\kappa_{ef} = \kappa_{eg} = 10L$. The bound on model Hessians κ_{bhm} can be chosen to be arbitrarily small, at the expense of limiting the class of models; however, it is clearly reasonable to choose κ_{bhm} as something that scales with L if this information is available. Let us assume that $\kappa_{bhm} = 10L$, as well. In a standard trust-region method, a common choice of algorithmic parameters would use $\kappa_{fcd} = \frac{1}{2}$, $\gamma = 2$, and $\eta_1 = \frac{1}{2}$.

The reader can verify that with these parameter choices and previous assumptions, Lemma 2.5.1 states that we must choose $\eta_2 \geq 32L$. *The intermediate constants satisfy* $\zeta \geq 42L$ *and* $C_1 = \frac{2}{17}$. *Without loss of generality, we will simply accept* $\zeta = 42L$.

Given the above values of the constants, we have

$$\max\left\{\frac{4}{\zeta C_1}, \frac{4}{\eta_1\eta_2\kappa_{fcd}}, \frac{1}{\kappa_{ef}}\right\} \leq \frac{1}{L},$$

and so

$$\frac{(\alpha\beta - \frac{1}{2})}{(1-\alpha)(1-\beta)} \geq 9 \tag{2.42}$$

and

$$(1-\alpha)(1-\beta) \leq \frac{1}{440}. \tag{2.43}$$

To generalize, supposing that κ_{ef}, κ_{eg}, *and* κ_{bhm} *scale linearly with* L, *then* η_2, ϵ_F, *and the expressions relating to* α *and* β *in Corollary 2.5.1 are all functions in* L *satisfying*

$$\eta_2 \geq \Theta(L), \tag{2.44}$$

$$\epsilon_F \leq \Theta(L), \tag{2.45}$$

$$\frac{\alpha\beta}{(1-\alpha)(1-\beta)} \geq \Theta(1), \tag{2.46}$$

and

$$(1-\alpha)(1-\beta) \leq \Theta(1), \tag{2.47}$$

where we use the notation $\Theta(\cdot)$ *to indicate the* $O(\cdot)$ *relationship with moderate constants.*

Remark 2.5.4. *Recall our remark made earlier in Theorem 2.5.2, case 2(b), on how* η_2 *bounds our step sizes. Indeed, if* $\eta_2 \geq \Theta(L)$ *has to be imposed this may force the algorithm to take small step sizes throughout. However, as mentioned earlier, the analysis of Theorem 2.5.2 can be modified by introducing a tradeoff between the size of* η_2 *and the accuracy parameters* ϵ_F *and* κ_{ef} *(as both of these constant parameters can be made smaller). It also may be advantageous to choose* η_2 *dynamically. Exploring this is a subject for future work.*

In the practical implementations that we will discuss in Section 6, we do not make use of the algorithmic parameter η_2 at all, and so even though η_2 is effectively arbitrarily close to 0, the algorithm still works.

Remark 2.5.5. *Note that if $\beta = 1$, then $\Delta_k \to 0$ for $\alpha \geq \frac{1}{2}$, which is the case shown in [4]. This is because, in our discussion of Case 1, the condition* (2.33) *via an appropriate adjustment to* (2.26) *could be written as*

$$[C_1\alpha\beta - (1-\alpha)(1-\beta)C_3] \geq \theta_1 C_1 \geq \theta_2 \frac{(1-\nu)(\gamma^2-1)}{\nu\zeta},$$

where θ_1 is positive and arbitrarily close to zero and $\theta_2 > 1$ is arbitrarily close to one. In the proof we provided, we chose values of $\theta_1 = \frac{1}{2}$ and $\theta_2 = 2$ for simplicity of the presentation.

2.5.1 The liminf-type convergence

We are ready to prove a liminf-type first-order convergence result, i.e., that a subsequence of the iterates of Algorithm 1 drive the gradient of the objective function to zero. The proof follows closely that given in [4], the key difference being the assumption on the quality of the function estimates.

Theorem 2.5.3. *Let the assumptions of Theorem 2.5.2 and Corollary 2.5.1 hold. Then the sequence of random iterates generated by Algorithm 1, $\{X_k\}$, almost surely satisfies*

$$\liminf_{k\to\infty} \|\nabla f(X_k)\| = 0.$$

Proof. We prove this result by contradiction conditioned on the almost sure event $\Delta_k \to 0$. Let us thus assume that there exists ϵ' such that, with positive probability, we have

$$\|\nabla f(X_k)\| \geq \epsilon', \quad \forall k.$$

Let $\{x_k\}$ and $\{\delta_k\}$ be realizations of $\{X_k\}$ and $\{\Delta_k\}$, respectively for which $\|\nabla f(x_k)\| \geq \epsilon'$ for all k. Since $\lim_{k\to\infty} \delta_k = 0$ (because we conditioned on $\Delta_k \to 0$), there exists k_0 such

that for all $k \geq k_0$,

$$\delta_k < b := \min\left\{\frac{\epsilon'}{2\kappa_{eg}}, \frac{\epsilon'}{2\kappa_{bhm}}, \frac{\kappa_{fcd}(1-\eta_1)\epsilon'}{16\kappa_{ef}}, \frac{\epsilon'}{2\eta_2}, \frac{\delta_{\max}}{\gamma}\right\}. \tag{2.48}$$

We define a random variable R_k with realizations $r_k = \log_\gamma\left(\frac{\delta_k}{b}\right)$. Then for the realization $\{r_k\}$ of $\{R_k\}$, $r_k < 0$ for $k \geq k_0$. The main idea of the proof is to show that such realizations occur only with probability zero, hence obtaining a contradiction with the initial assumption of $\|\nabla f(x_k)\| \geq \epsilon'$ for all k.

We first show that R_k is a submartingale. Recall the events I_k and J_k in Definitions 2.4.2 and 2.4.5. Consider some iterate $k \geq k_0$ for which I_k and J_k both occur, which happens with probability $P(I_k \cap J_k) \geq \alpha\beta$. Since (2.48) holds, we have exactly the same situation as in Case 1(a) in the proof of Theorem 2.5.2. In other words, we can apply Lemmas 2.5.2 and 2.5.3 to conclude that the k-th iteration is successful, hence, the trust-region radius is increased. In particular, since $\delta_k \leq \frac{\delta_{\max}}{\gamma}$, $\delta_{k+1} = \gamma\delta_k$. Consequently, $r_{k+1} = r_k + 1$.

Let $\mathcal{F}_{k-1}^{I\cdot J} = \sigma(I_0, \cdots, I_{k-1}) \cap \sigma(J_0, \cdots, J_{k-1})$. For all other outcomes of I_k and J_k, which occur with total probability of at most $1 - \alpha\beta$, we have $\delta_{k+1} \geq \gamma^{-1}\delta_k$. Hence

$$\mathbb{E}[r_{k+1}|\mathcal{F}_{k-1}^{I\cdot J}] \geq \alpha\beta(r_k + 1) + (1 - \alpha\beta)(r_k - 1) \geq r_k,$$

because $\alpha\beta > 1/2$ as a consequence of the assumptions from Corollary 2.5.1. This implies that R_k is a submartingale.

Now let us construct another submartingale, W_k, on the same probability space as R_k which will serve as a lower bound on R_k and for which $\left\{\limsup_{k\to\infty} W_k = \infty\right\}$ holds almost surely. Define indicator random variables $\mathbf{1}_{I_k}$ and $\mathbf{1}_{J_k}$ such that $\mathbf{1}_{I_k} = 1$ if I_k occurs, $\mathbf{1}_{I_k} = 0$ otherwise, and similarly, $\mathbf{1}_{J_k} = 1$ if J_k occurs, $\mathbf{1}_{J_k} = 0$ otherwise. Then define

$$W_k = \sum_{i=0}^{k}(2 \cdot \mathbf{1}_{I_k} \cdot \mathbf{1}_{J_k} - 1).$$

Notice that W_k is a submartingale since

$$\begin{aligned}
\mathbb{E}[W_k|\mathcal{F}_{k-1}^{I\cdot J}] &= E[W_{k-1}|\mathcal{F}_{k-1}^{I\cdot J}] + E[2\cdot \mathbf{1}_{I_k}\cdot \mathbf{1}_{J_k} - 1|\mathcal{F}_{k-1}^{I\cdot J}] \\
&= W_{k-1} + 2E[\mathbf{1}_{I_k}\cdot \mathbf{1}_{J_k}|\mathcal{F}_{k-1}^{I\cdot J}] - 1 \\
&= W_{k-1} + 2P(I_k \cap J_k|\mathcal{F}_{k-1}^{I\cdot J}) - 1 \\
&\geq W_{k-1},
\end{aligned}$$

where the last inequality holds because $\alpha\beta \geq 1/2$. Since W_k only has ± 1 increments, it has no finite limit. Therefore, by Theorem 2.5.1, we have $\left\{\limsup_{k\to\infty} W_k = \infty\right\}$.

By the construction of R_k and W_k, we know that $r_k - r_{k_0} \geq w_k - w_{k_0}$. Therefore, R_k is positive infinitely often with probability one. This implies that the sequence of realizations r_k such that $r_k < 0$ for $k \geq k_0$ occurs with probability zero. Therefore our assumption that $\|\nabla f(X_k)\| \geq \epsilon'$ holds for all k with positive probability is false and

$$\liminf_{k\to\infty} \|\nabla f(X_k)\| = 0$$

holds almost surely.

□

2.5.2 The lim-type convergence

In this subsection we show that $\lim_{k\to\infty} \|\nabla f(X_k)\| = 0$ almost surely.

We now state an auxiliary lemma, which is similar to the one in [4], but requires a different proof since here the function values $f(X_k)$ can increase with k, while in the case considered in [4], function values are monotonically nonincreasing.

Lemma 2.5.5. *Let the same assumptions that were made in Theorem 2.5.3 hold. Let $\{X_k\}$ and $\{\Delta_k\}$ be sequences of random iterates and random trust-region radii generated by Algorithm 1. Fix $\epsilon > 0$ and define the sequence $\{K_\epsilon\}$ consisting of the natural numbers*

k for which $\|\nabla f(X_k)\| > \epsilon$ *(note that* K_ϵ *is a sequence of random variables). Then,*

$$\sum_{k \in \{K_\epsilon\}} \Delta_k \; < \; \infty$$

almost surely.

Proof. From Theorem 2.5.2 we know that $\sum \Delta_k^2 < \infty$ and hence $\Delta_k \to 0$ almost surely. For each realization of Algorithm 1 and a sequence $\{\delta_k\}$, there exists k_0 such that $\delta_k \leq \epsilon/\zeta$, $\forall k \geq k_0$, where ζ is defined as in Theorem 2.5.2. Let K_0 be the random variable with realization k_0 and let K denote the sequence of indices k such that $k \in K_\epsilon$ and $k \geq K_0$. Then for all $k \in K$, Case 1 of Theorem 2.5.2 holds, i.e., $\|\nabla f(X_k)\| \geq \zeta \Delta_k$, since $\|\nabla f(X_k)\| \geq \epsilon$ for all $k \in K$. From this and from (2.34) we have

$$E[\Phi_{k+1} - \Phi_k | \mathcal{F}_{k-1}^{M \cdot F}] \leq -\frac{1}{4} C_1 \nu \epsilon \Delta_k, \quad \forall k \geq k_0.$$

Recall that Φ_k is bounded from below. Hence, summing the above inequality for all $k \in K$ and taking the expectation, we have that

$$\sum_{k \in K} \Delta_k \; < \; \infty$$

almost surely. Since $K_\epsilon \subseteq K \cup \{k : k \leq K_0\}$ and K_0 is finite almost surely, the lemma follows. □

We are now ready to state the lim-type result.

Theorem 2.5.4. *Let the same assumptions as in Theorem 2.5.3 hold. Let* $\{X_k\}$ *be a sequence of random iterates generated by Algorithm 1. Then, almost surely,*

$$\lim_{k \to \infty} \|\nabla f(X_k)\| \; = \; 0.$$

Proof. The proof of this result is almost identical to the proof of the same theorem in [4]; hence we will not present the proof here. The key idea of the proof is to show that if the

theorem does not hold, then with positive probability

$$\sum_{k \in \{K_\epsilon\}} \Delta_k \;=\; \infty, \tag{2.49}$$

with K_ϵ defined as in Lemma 2.5.5. This result is shown using Lipschitz continuity of the gradient and does not depend on the stochastic nature of the algorithm. Since (2.49) contradicts the almost sure result of Lemma 2.5.5, the theorem is proven. □

2.6 Constructing models and estimates in different stochastic settings.

We now discuss various settings of stochastic noise in the objective function and how α-probabilistically κ-fully linear models and β-probabilistically ϵ_F-accurate estimates can be obtained in these settings.

Recall that we assume that for each x we can compute the value of $\tilde{f}(x)$, which is the noisy version of f,

$$\tilde{f}(x) = f(x, \omega),$$

where ω is a random variable inducing the noise.

Simple stochastic noise. Let us first consider a somewhat general setting of unbiased noise, i.e.,

$$E_\omega[f(x, \omega)] = f(x), \quad \forall x,$$

and

$$\mathrm{Var}_\omega[f(x, \omega)] \leq V < \infty, \quad \forall x.$$

This is a typical noise assumption in stochastic optimization literature. In the case of unbiased noise as above, constructing estimates and models that satisfy our assumptions is fairly straight-forward. First, let us consider the case where only noisy zeroth-order function estimates are available (i.e. gradient information is unavailable), and we want to construct a model that is κ-fully linear in a given trust region $B(x^0, \delta)$ with some sufficiently

high probability, α.

One can employ standard SAA techniques to reduce the variance of the function estimates. In particular, let $\bar{f}_p(x,\omega) = \frac{1}{p}\sum_{i=1}^{p} f(x,\omega_i)$, where ω_i are the i.i.d. realizations of the noise ω. Then, by Chebyshev inequality, for any $v > 0$,

$$P(|\bar{f}_p(x,\omega) - f(x)]| > v) = P(|\bar{f}_p(x,\omega) - E_\omega[f(x,\omega)]| > v) \leq \frac{V}{pv^2}.$$

In particular, we want $v = \kappa'_{ef}\delta^2$ for some $\kappa'_{ef} > 0$ and $\frac{V}{pv^2} \leq 1 - \alpha'$ for some α', which can be ensured by choosing $p \geq \frac{V}{(\kappa'_{ef})^2(1-\alpha')\delta^4}$.

We now construct a fully linear model as follows: given a well-poised set[3] Y of $n+1$ points in $B(x^0,\delta)$, at each point $y^i \in Y$, we compute $\bar{f}_p(y^i,\omega)$ and build a linear interpolation model $m(x)$ such that $m(y^i) = \bar{f}_p(y^i,\omega)$, for all $i = 1,\ldots,n+1$. Hence, for any $y^i \in Y$, we have

$$P(|m(y^i) - f(y^i)]| > \kappa'_{ef}\delta^2) \leq 1 - \alpha'.$$

Moreover, the events $\{|m(y^i) - f(y^i)]| > \kappa'_{ef}\delta^2\}$ are independent, hence

$$P(\max_{i=1..n+1}\{|m(y^i) - f(y^i)]|\} > \kappa'_{ef}\delta^2) \leq 1 - (\alpha')^{n+1}.$$

It is easy to show using, for example, techniques described in [21], that $m(x)$ is a κ-fully linear model of $E_\omega[f(x,\omega)]$ in $B(x^0,\delta)$ for appropriately chosen $\kappa = (\kappa_{eg},\kappa_{ef})$, with probability at least $\alpha = (\alpha')^{n+1}$.

Computing the β-probabilistically ϵ_F-accurate estimates of $f(x,\omega)$ can be done analogously to the construction of the models described above.

The majority of stochastic optimization and SAA methods focus on derivative-based optimization where it is assumed that, in addition to $f(x,\omega)$, $\nabla_x f(x,\omega)$ is also available,

[3]See [21] for details on well-poised sets and how they can be obtained. We will approach this topic in much further detail in Chapter 4.

and that the noise in the gradient computation is also independent of x, that is [4]

$$E_\omega[\nabla_x f(x,\omega)] = \nabla f(x), \quad \forall x$$

and

$$\mathrm{Var}_\omega[\|\nabla_x f(x,\omega)\|] \leq V < \infty, \quad \forall x.$$

In the case where noisy gradient values are available, the construction of fully linear models in $B(x^0,\delta)$ is simpler. Let $\bar{\nabla} f_p(x,\omega) = \frac{1}{p}\sum_{i=1}^{p} \nabla f(x,\omega_i)$. Again, by an extension of the Chebyshev inequality, for p such that

$$p \geq \max\{\frac{V}{\kappa_{ef}^2(1-\alpha')\delta^4}, \frac{V}{\kappa_{eg}^2(1-\alpha')\delta^2}\},$$

$$P(\|\bar{\nabla} f_p(x^0,\omega) - \nabla f(x^0)]\| > \kappa_{eg}\delta) = P(\|\bar{\nabla} f_p(x^0,\omega) - E_\omega[\nabla f(x^0,\omega)]\| > \kappa_{eg}\delta) \leq 1-\alpha',$$

and

$$P(|\bar{f}_p(x^0,\omega) - f(x^0)| > \kappa_{ef}\delta^2) = P(|\bar{f}_p(x^0,\omega) - E_\omega[f(x^0,\omega)]| > \kappa_{ef}\delta^2) \leq 1-\alpha'.$$

Hence the linear expansion $m(x) = \bar{f}_p(x^0,\omega) + \bar{\nabla} f_p(x^0,\omega)^\top (x - x^0)$ is a κ-fully linear model of $f(x) = E_\omega[f(x,\omega)]$ on $B(x^0,\delta)$ for appropriately chosen $\kappa = (\kappa_{eg}, \kappa_{ef})$, with probability at least $\alpha = (\alpha')^2$.

There are many existing SG and SAA methods with convergence guarantees for stochastic optimization with i.i.d. or unbiased noise. Some of these methods have been shown to achieve the optimal sampling rate [35], i.e. they converge to the optimal solution while sampling the gradient at the best possible rate. We will explore convergence rates further in Chapter 3. Our contribution in this chapter is a method which applies in non-i.i.d. noise regimes, as we will discuss below. In Section 2.7, however, we will demonstrate that simple implementations of STORM can have superior numerical behavior compared to more standard SAA approaches, even in i.i.d. noise regimes, so the idea is at least competitive

[4] In general, the variance of the gradient estimate and the function estimate are not the same, but for simplicity, we bound both here by V.

in practice.

Function computation failures. Now, let us consider a more complex noise case. Suppose that the function $f(x)$ is computed (as it often is, in black-box optimization) by some numerical method that includes a random component. Such examples are common in machine learning applications, for instance, where x is a vector of hyperparameters of a learning algorithm and $f(x)$ is the expected error of the resulting classifier. In this case, for each value of x, the classifier is obtained by solving an optimization problem, up to a certain accuracy, on a given training set. If a randomized coordinate descent algorithm or a stochastic gradient descent algorithm is used to train the classifier for a given vector of hyperparameters x, then the resulting classifier is sufficiently close to the optimal classifier with some known probability under certain conditions, e.g. strong convexity of a loss function. However, in the case where the training optimizer fails to produce a sufficiently accurate solution, the resulting error is difficult to estimate. Usually, it is possible to know the upper bound on the value of this inaccurate objective function but nothing else may be known about the distribution of this value. Moreover, it is likely that the probability of obtaining an accurate estimate of $f(x)$ depends on x; for example, after k iterations of randomized coordinate descent, the error between the true $f(x)$ and the computed $\tilde{f}(x)$ is bounded by some value, with probability α, where the value depends on α, k, and x [63].

Another example is solving a system of nonlinear black-box equations. Assume that we seek x such that $\sum_i (f_i(x))^2 = 0$, for some functions $f_i(x)$, $i = 1, \ldots, m$ that are computed by numerical simulation, with noise. As is often done in practice (and is supported by our theory) the noise in the function estimate is reduced as the algorithm progresses, for example, by reducing the size of a discretization or convergence tolerance within the black-box computation. These adjustments for noise reduction usually increase the workload of the simulation. With the increase of the workload, there is an increased probability of failure of the black-box code. Hence, as the sum of the squares of estimates of $f_i(x)$ becomes smaller, the more likely it becomes that the computation of the estimated sum will fail and some inaccurate value is returned.

These two examples show that the noise in $\tilde{f}(x)$ may be large with some positive

probability, which may depend on x. Hence, let us consider the following, idealized, noise model

$$\tilde{f}(x) = f(x, \omega) = \begin{cases} f(x), & \text{w.p. } 1 - \sigma(x) \\ \omega(x) \leq V & \text{w.p. } \sigma(x), \end{cases}$$

where $\sigma(x)$ is the probability with which the function $f(x)$ is computed inaccurately, and $\omega(x)$ is some random function of x, for which only an upper bound V is known. This case is idealized, because we assume that with probability $1 - \sigma(x)$, $f(x)$ is computed exactly. It is trivial to extend this example to the case when $f(x)$ is computed with an error, but this error can be made sufficiently small.

For this model of function computation failures we have

$$E_\omega[f(x, \omega)] = (1 - \sigma(x))f(x) + \sigma(x)E[\omega(x)] \neq f(x), \quad \forall \sigma(x) > 0.$$

and it is clear, that for any $\sigma(x) > 0$, unless $E[\omega(x)] \equiv$ some constant, optimizing $E_\omega[f(x, \omega)]$ does not give the same result as optimizing $f(x)$. Hence applying Monte-Carlo sampling within an optimization algorithm solving this problem is not a correct approach.

We now observe that constructing α-probabilistically κ-fully linear models and β-probabilistically ϵ_F-accurate estimates is trivial in this case, assuming that $\sigma(x) \leq \sigma$ for all x, when σ is small enough. In particular, given a trust region $B(x^0, \delta)$, sampling a function $f(x)$ on a sample set $Y \subset B(x^0, \delta)$ well-poised for linear interpolation will produce a κ-fully linear model in $B(x^0, \delta)$ with probability at least $(1-\sigma)^{|Y|}$, since with this probability all of the function values are computed exactly. Similarly, for any $s \in B(x^0, \delta)$, the function estimates F^0 and F^s are both correct with probability at least $(1 - \sigma)^2$. Assuming that $(1 - \sigma)^{|Y|} \geq \alpha$ and $(1 - \sigma)^2 \geq \beta$, where α and β satisfy the assumptions of Theorem 2.5.2 and Lemma 2.5.1 and $\alpha\beta \geq \frac{1}{2}$ as in Theorem 2.5.3, we observe that the resulting models satisfy our theory.

Remark 2.6.1. *We assume here that the probability of failure to compute $f(x)$ is small enough for all x. In the machine learning example provided above, it is often possible to control the probability $\sigma(x)$ in the computation of $f(x)$, for example by increasing the*

number of iterations of a randomized coordinate descent or stochastic gradient descent method. In the case of the black-box nonlinear equation solver, the probability of code failure is expected to be quite small. There are, however, examples of black box optimization problems where the computation of $f(x)$ fails all the time for specific values of x. This is often referred to as hidden constraints [55]. Clearly our theory does not apply here, but we believe there is no local method that can provably converge to a local minimizer in such a setting without additional information about these specific values of x.

2.7 Computational Experiments

In this section, we will discuss the performance of several variants of our proposed method (varied in the way the models are constructed), henceforth only referred to as STORM (STochastic Optimization using Random Models), that target various noisy situations discussed in the previous section. We note that a comparison of STORM to the SPSA method of [70] and the classical Kiefer-Wolfowitz method in [42] has been reported in [16] and shows that STORM significantly outperformed these two methods, while no special tuning of SPSA or Kiefer-Wolfowitz was applied. Since a trust-region based method, which is able to use second-order information, is likely to outperform stochastic gradient-like methods in many settings, we omit such comparison here.

Throughout this section, all proposed algorithms were implemented in Matlab and all experiments were performed on a laptop computer running Ubuntu 14.04 LTS with an Intel Celeron 2955U @ 1.40GHz dual processor.

2.7.1 Simple stochastic noise

In these experiments, we used a set of 53 unconstrained problems adapted from the CUTEr test set, each being in the form of a sum of squares problem, i.e.

$$f(x) = \sum_{i=1}^{m} (f_i(x))^2, \tag{2.50}$$

where for each $i \in \{1, \ldots, m\}$, $f_i(x)$ is a smooth function. Two different types of noise will be used in this first subsection, which we will refer to as *multiplicative* noise and *additive* noise. In the multiplicative noise case, for each $i \in \{1, \ldots, m\}$, we generate some ω_i from the uniform distribution on $[-\sigma, \sigma]$ for some parameter $\sigma > 0$, and then compute the noisy function

$$\tilde{f}(x, \omega) = \sum_{i=1}^{m} ((1 + \omega_i) f_i(x))^2. \tag{2.51}$$

The key characteristic of this noise is that for each x, we have $E_\omega[f(x, \omega)] = f(x)$, however the variance is nonconstant over x and scales with the magnitudes of the components $f_i(x)$. Thus, one should expect that if an algorithm is minimizing a function of the form (2.51), the accuracy of the estimates of $f(x)$ based on a constant number of samples of $\tilde{f}(x, \omega)$ should increase, assuming that the algorithm produces a decreasing sequence $\{f(x^k)\}_{k=1}^{\infty}$. While this behavior is not supported by theory, because we do not know how quickly $f(x^k)$ decreases, our computational results show that a constant number of samples is indeed sufficient for convergence.

The other type of noise we will test is *additive*, i.e. we additively perturb each component in (2.50) by some ω_i uniformly generated in $[-\sigma, \sigma]$ for some parameter $\sigma > 0$. That is,

$$\tilde{f}(x, \omega) = \sum_{i=1}^{m} (f_i(x) + \omega_i)^2 \tag{2.52}$$

Note that the noise is additive only in terms of the component functions, but not in terms of the objective function. Moreover, $E_\omega[f(x, \omega)] = f(x) + \sum_i^m E(\omega_i)^2$. However, the constant bias term does not affect optimization results, since $\min_x E_\omega[f(x, \omega)] = \min_x f(x)$.

In our first set of experiments for these two noisy settings, we compare a version of STORM to a version of a SAA-based trust-region algorithm, which we will call "TR-SAA", which is similar to a trust-region algorithm presented in [24]. Another similar method, with convergence guarantees, has been recently proposed in [68]. In their work,

they use a Bayesian scheme to select a sufficiently large sample complexity for computing average function values at a current interpolation set. Here, in TR-SAA, we simplify this approach, by increasing sample complexity in each iteration proportionally to the decrease of the trust region radius δ_k. A description of TR-SAA is given in Algorithm 6 in the Appendix.

There are two particular aspects of TR-SAA that we would like to draw attention to: in the estimate calculation step, the computation of f_k^0 is performed *before* the model m_k is constructed, and m_k is built to interpolate f_k^0; hence the quality of estimate f_k^0 and that of the model m_k are *not independent*. Additionally, the quality of the model m_k is dependent on that of m_{k-1} because the function estimates are reused. Both of these aspects are violations of the guiding assumptions of STORM. Thus, we also propose TR-SAA-resample, which is the same algorithm as TR-SAA except that at each iteration, the function estimate at every interpolation point is recomputed as an average of function evaluations, independent of past function evaluations. While TR-SAA-resample may overcome TR-SAA's issue of the dependence of m_k on m_{k-1}, it still doesn't satisfy the assumptions of STORM because of the dependence of f_k^0 on m_k.

Thus, finally in Algorithm 7, stated in the Appendix, we propose a version of STORM, STORM-unbiased, comparable to TR-SAA in terms of per-iteration sample complexity. In Algorithm 7, the models m_k and m_{k-1} are entirely independent since a new regression set is drawn in each iteration. Additionally, in the estimates calculation step, the computations of f_k^0 and f_k^s are completely independent of the model m_k. For these reasons, Algorithm 7 is more in line with the theory analyzed in this chapter than the SAA scheme given in Algorithm 6. We will further comment on the quality of least-squares regression models in Algorithm 7 in great detail in Chapter 4.

For each of the 53 problems, the best-known value of the noiseless $f(x)$ obtained by a solver is recorded as f^*. We recorded the number of function evaluations required by a solver to obtain a function value $f(x^k) < f'$ such that

$$1 - \tau < \frac{f(x^0) - f'}{f(x^0) - f*}. \tag{2.53}$$

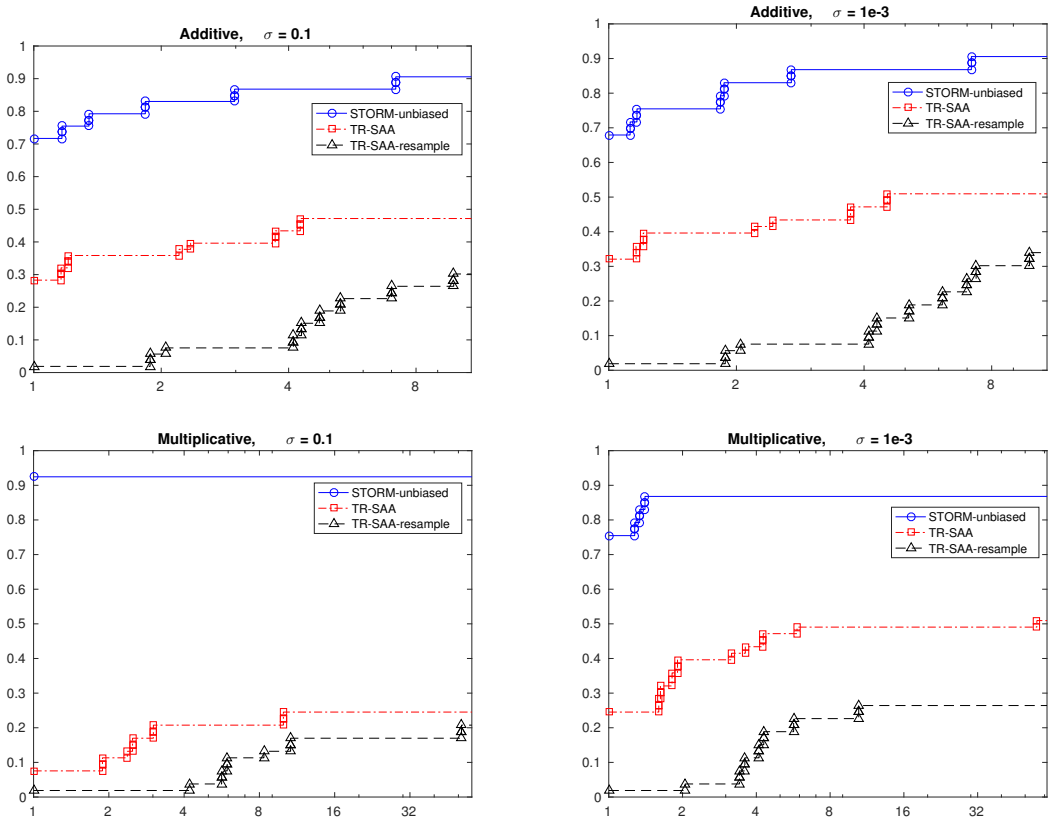

Figure 2.1: Performance profiles ($\tau = 10^{-3}$) comparing STORM-unbiased, TR-SAA, and TR-SAA-resample on the testset.

This number was averaged over 10 runs for each problem. In the profiles shown in Figure 2.7.1, $\tau = 10^{-3}$. In all the experiments, a budget of $1000(n+1)$ noisy function evaluations was set, where n is the problem dimension. For the choice of initialization, the same parameters were used in all of TR-SAA, TR-SAA-resample, and STORM-unbiased: $\delta_{\max} = 10, \delta_0 = 1, \gamma = 2, \eta_1 = 0.1, \eta_2 = 0.001, p_{\min} = 10$.

Note that even though we have ignored the theoretical prescription derived in Section 2.6 that the sample rate should scale with $1/\delta_k^4$, we note that STORM-unbiased performs extremely well compared to the TR-SAA method. Although we chose to sample at a rate so that p_k was on the order of $1/\delta_k$, this particular sample rate was chosen after testing various other rates on the same set of test functions, and seemed to work relatively well for both STORM-unbiased and TR-SAA.

Function computation failures. In these experiments, we used the same 53 sum-of-squares problems as in the unbiased noise experiments described above, but introduced biased noise. For each component in the sum in (2.50), if $|f_i(x)| < \epsilon$ for some parameter $\epsilon > 0$, then $f_i(x)$ is computed as

$$f_i(x) = \begin{cases} f_i(x) & \text{w.p. } 1 - \sigma \\ V & \text{w.p. } \sigma \end{cases}$$

for some parameter $\sigma > 0$ and for some "garbage value" V. If $f_i(x) \geq \epsilon$, then it is deterministically computed as $f_i(x)$. This noise is biased, with bias depending on x, and as such we should not expect any sort of averaging to work well here. Once the garbage value V was chosen with sufficiently large magnitude, the relative magnitude of V did not significantly affect the results, and so in the experiments illustrated below $V = 10000$ was used. Obviously, the intention here was that such a large value will cause STORM to see a trial step as unacceptable, when it may, in fact, decrease the true function value if taken. We propose the version of STORM presented as Algorithm 8 in the Appendix.

The key feature of Algorithm 8 is that on each iteration, the interpolation set changes minimally as in a typical DFO trust region method, but the interpolated function values are computed afresh. Intuitively, this is the right thing to do, since if a "garbage value" is computed at some point in the algorithm to either construct a model or provide a function value estimate, we do not want its presence to affect the computation of models in subsequent iterations. No averaging is performed, as it can only cause harm in this setting.

Since we are not aware of any other optimization algorithm that is designed for this noise regime, we performed no comparisons, but experiment to discover how the method works as a function of the probability of failure on our test set. On the test set of 53 problems, we ran Algorithm 8 a total of 30 times and report the average percentage of the 53 instances that are solved in the sense of (2.53) with $\tau = 10^{-3}$ within a budget of $10000(n+1)$ function evaluations, where f^* was computed by Algorithm 8 with $\sigma = 0$. In order to standardize the probability of failure over the test set, we define the probability of success p and then take σ on a function with m components as $\sigma = 1 - p^{(1/m)}$. The results

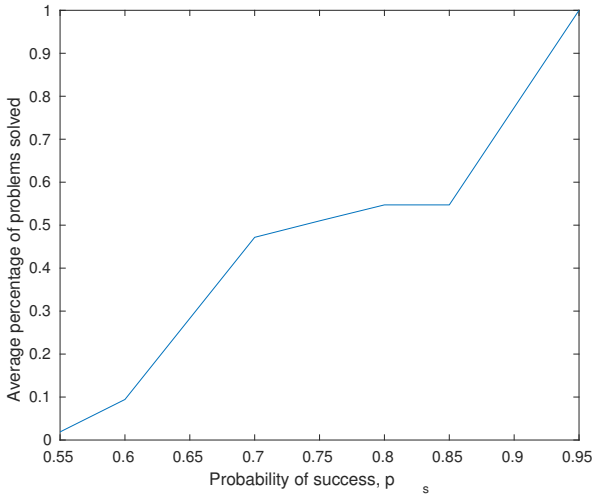

Figure 2.2: Average percentage of problems solved as a function of probability of success p.

are summarized in Figure 2.7.1.

This experiment suggests that the practical threshold at which STORM fails to make progress may be looser than that suggested by theory. We will illustrate this idea through a simple example. Consider the minimization of the simple quadratic function

$$f(x) = \sum_{i=1}^{n} (x_i - 1)^2. \tag{2.54}$$

The minimizer uniquely occurs at the vector of all 1s. Now consider the minimization of this function under our setting of computation failure where we vary the probability parameter σ and fix $\epsilon = 0.1$. Suppose on each iteration of STORM, the interpolation set contains $(n+1)(n+2)/2$ points. Then, the probability of obtaining the correct quadratic model in the worst case where for all i, $|x_i - 1| < \epsilon$ is precisely $\alpha = ((1-\sigma)^n)^{\frac{(n+1)(n+2)}{2}}$. Likewise, the probability of obtaining the correct function evaluation for F_0 and F_s on each iteration in the worst case is $\beta = ((1-\sigma)^n)^2$. Now, supposing we initialize STORM with the zero vector in $\mathbb{R}^n$, it is reasonable to assume that all iterates will occur near the unit cube $[0,1]^n$, and so we can use simple calculus to estimate a Lipschitz constant of the function over the relevant domain as $2\sqrt{n}$, and the Lipschitz constant of the gradient is constantly 2. These also serve as reasonable estimates of κ_{ef} and κ_{eg}, respectively. Using

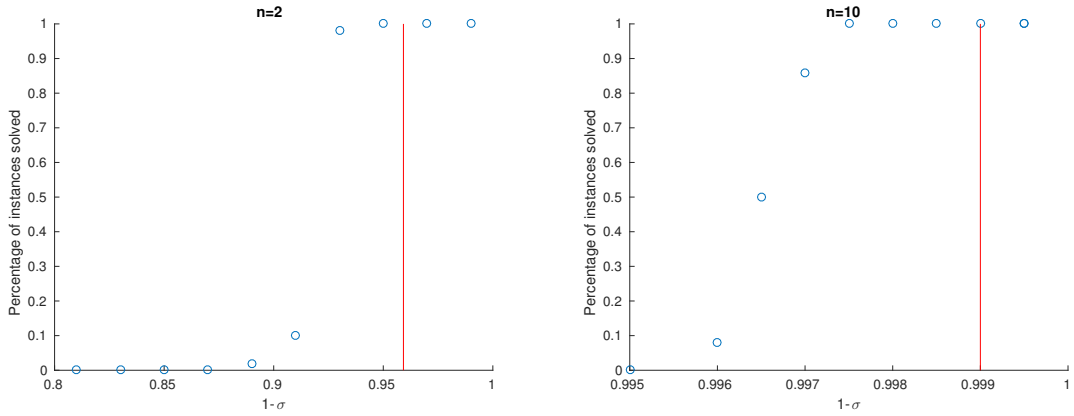

Figure 2.3: Minimizing a simple quadratic function (dimension $n = 2$ in the left, $n = 10$ in the right) where with probability $1 - \sigma$, a coordinate is computed incorrectly.

parameter choices $\gamma = 2$, $\eta_1 = 0.1$, $\eta_2 = 1$, we can use the bounds in (2.40) and (2.41) to solve for the smallest allowable $(1-\sigma)$ for which our algorithm can guarantee convergence. For $n = 2$, our theory suggests that we can safely lower bound $(1 - \sigma) > 0.9592$ (which implies $\alpha \approx 0.6069$ and $\beta \approx 0.8467$), and for $n = 10$, we can lower bound $(1 - \sigma) > 0.9990$ (which implies $\alpha \approx 0.5233$ and $\beta \approx 0.9806$).

In Figure 2.7.1 for $n = 2, 10$, we plot an indicated level of $(1 - \sigma)$ on the x-axis against the proportion of 100 randomly-seeded instances with that level of $(1 - \sigma)$ that Algorithm 8 managed to find a solution x^* satisfying $f(x^*) < 10^{-5}$ within 10^4 many function evaluations using the discussed parameter choices. The red line shows the level of $(1 - \sigma)$ that our theory predicted in the previous paragraph. As we can see, $(1 - \sigma)$ can be quite a bit smaller than predicted by our theory before the failure rate becomes unsatisfactory. As a particular example, in the $n = 10$ case, when $(1 - \sigma) = .998$, the corresponding probabilities are $\alpha \approx 0.266782$ and $\beta \approx 0.960751$, and yet 100% of the instances were solved to the required level of accuracy. In other words, even though the models are eventually only accurate on roughly 27% of the iterations, we still see satisfactory performance.

2.7.2 Stochastic gradient based method comparison

In this subsection, we show how STORM applies to empirical risk minimization in machine learning. We will consider this in greater detail later in Chapter 6. Consider a training dataset of N samples, $\{(x_i, y_i)\}_{i=1\ldots N}$, where $x_i \in \mathbb{R}^m$ is a vector of m real-valued and $y_i \in \{-1, 1\}$ indicates a positive or negative label respectively. We will train a linear classifier and bias term $(w, \beta) \in \mathbb{R}^{m+1}$ by minimizing the smooth (convex) regularized logistic loss

$$f(w,\beta) = \frac{1}{N}\sum_{i=1}^{N} f_i(w,\beta) + \lambda\|w\|^2 = \frac{1}{N}\sum_{i=1}^{N} \log(1+\exp(-y_i(w^T x_i + \beta))) + \lambda\|w\|^2.$$

As in the typical machine learning setting, we will assume that $N >> m$, and so computing $f(w,\beta)$, $\nabla f(w,\beta)$, and $\nabla^2 f(w,\beta)$ exactly are prohibitive. Hence we will only compute estimates of these quantities by considering a sample $I \subset \{1,\ldots,N\}$ of size $|I| = n << N$, yielding

$$f_I(w,\beta) = \frac{1}{|I|}\sum_{i\in I} f_i(w,\beta) + \lambda\|w\|^2,\ \nabla f_I(w,\beta) = \frac{1}{|I|}\sum_{i\in I} \nabla f_i(w,\beta) + 2\lambda w,$$

$$\nabla^2 f_I(w,\beta) = \frac{1}{|I|}\sum_{i\in I} \nabla^2 f_i(w,\beta) + 2\lambda I_n.$$

Thus we can construct models based on sample gradient and Hessian information and we present the appropriate variant of STORM as Algorithm 9 in the Appendix.

We compare Algorithm 9 with an implementation of the well-known Adagrad method from the Ada-whatever package [26]. We compare against this particular solver because it is a well-understood stochastic gradient method used by the machine learning community that, like our algorithm, takes adaptive step sizes, but unlike our algorithm, does not compute estimates of the loss function, but only computes averaged stochastic gradients. For the choice of initialization in Algorithm 9, the following parameters were used: $\delta_{\max} = 10, \delta_0 = 1, x_0 = 0, \gamma = 2, \eta_1 = 0.1, \eta_2 = 0.001, p_{\min} = m + 2, p_{\max} = N$. Adagrad

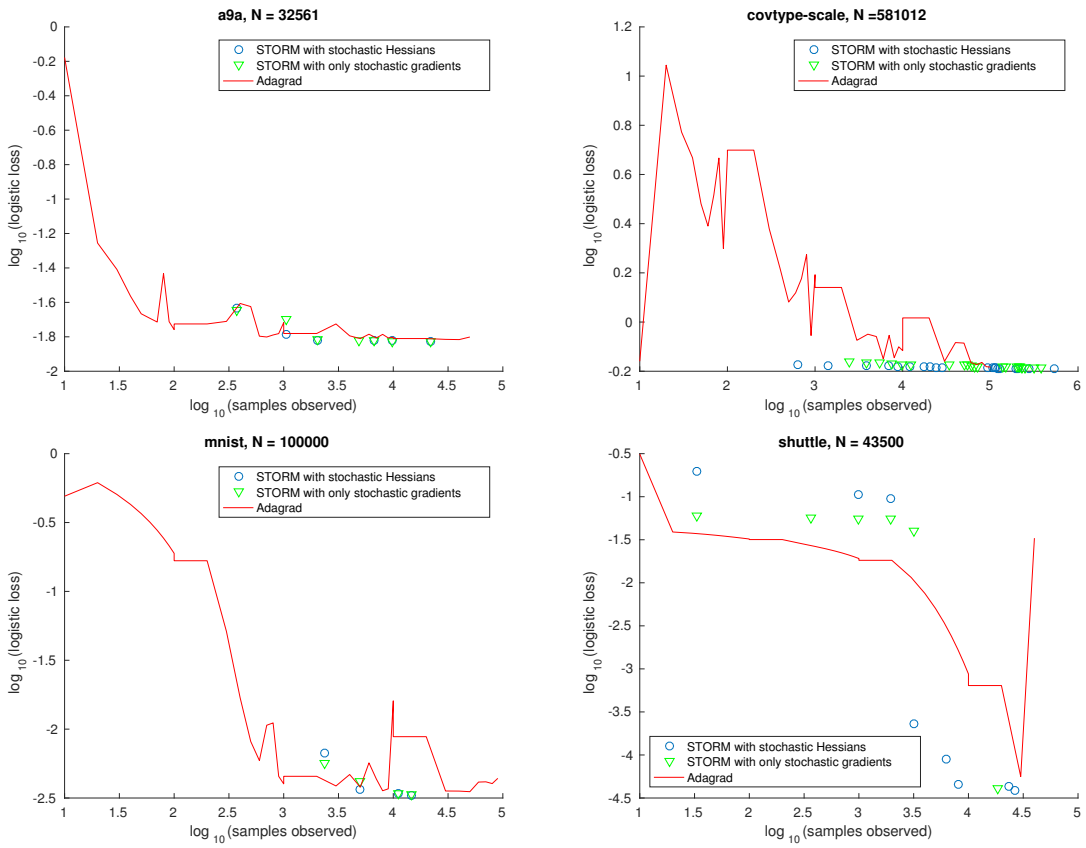

Figure 2.4: Trajectory of training logistic loss on four datasets.

was also given the same initial point and an initial step size of $\delta_0 = 1$. We remark that in practice, one would need to tune the parameters of Adagrad a bit for a particular problem, but for a fair comparison, we chose only to run Adagrad on these default settings.

We implemented two versions of Algorithm 9: one which uses stochastic Hessians, and a second where we do not compute stochastic Hessians, effectively setting $H_k = 0$ on each iteration, yielding a trivial subproblem in the step calculation.

We set the maximum budget of data evaluations for each solver equal to the size of the training set, thus comparing various solvers' performance with a budget of roughly one full pass through the dataset. For the two implementations of STORM, we plot in Figure 2.4 the true training loss function value at the end of each successful iteration, while for Adagrad, we simply plot the true training loss function value over an evenly spaced array of function evaluation counts.

Notice that, as expected, the true function values produced by Adagrad can vary widely

over this horizon, but implementations of STORM tend to yield fairly stable decreasing trajectories over its successful iterations.

Chapter 3

Theoretical Convergence Rate of STORM

3.1 Introduction

Having introduced the STORM algorithmic framework in Chapter 2, we now analyze its theoretical convergence rate[1]. Particularly driven by applications in machine learning (ML) domains, where stochastic gradient descent (SGD) arguably remains the method of choice even for non-convex problems, there has been an interest in convergence rates for SGD on non-convex problems. However, theoretical convergence rate results for SGD methods on non-convex problems have lagged behind the theoretical results for convex problems, for which SGD methods were primarily intended. A notable paper [33] is the first to provide convergence rates guarantees of some sort for a randomized stochastic gradient method in a non-convex setting. The particular method that they analyze, however, utilizes a carefully chosen step size and a randomized stopping scheme, which are quite different from what is used in practice. Additional work in the general ML task of minimizing a (possibly) very large sum of non-convex functions has led to analysis of stochastic variance-reduced gradient (SVRG) methods [40], for which convergence rates have been proven when the stochastic gradients are at least supposed to be Lipschitz continuous [1, 61].

[1]The work in this Chapter is joint work with Jose Blanchet, Coralia Cartis, and Katya Scheinberg. This Chapter is a distillation of the key ideas found in [9]

Rather than limiting ourselves to an objective consisting of a large sum of functions in this chapter, we consider the same general stochastic unconstrained, possibly non-convex, optimization problem from (2.1)

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

where $f(x)$ is a function which is assumed to be smooth and bounded from below, and whose value can only be computed with some noise. As before, let $\tilde{f}$ be the noisy computable version of f, which takes the form $\tilde{f}(x) = f(x, \xi)$ where the noise ξ is a random variable. Note that, as it was unnecessary in the convergence analysis of STORM, we do not make the typical unbiasedness assumptions that $f(x) = \mathbb{E}_\xi[f(x, \xi)]$ and $\text{Var}[f(x, \xi)] < \sigma^2$. Instead, throughout the analysis in this chapter, we will continue to assume as in Chapter 2 that at each iteration, we can obtain a sufficiently accurate model and pair of function estimates with a sufficiently high probability p, conditioned on the past. We can interpret p as $p = \alpha\beta$, where α and β are from the definitions of α-probabilistically fully-linear models and β-probabilistically ϵ_F-accurate estimates, respectively. This means that we allow (arbitrary) errors in function (and possibly gradient) estimates with some small probability throughout the algorithm. We show that the expected number of iterations required to achieve $\|\nabla f(x)\| \leq \epsilon$ is bounded by $O(\epsilon^{-2}/(2p-1))$, which is an improvement on the result in [33] and a similar result as in [1], in terms of dependence on ϵ. However, in the specific form of the sum-of-functions problem analyzed in [1], the occasional computation of a "full gradient" is performed; since such an object may not be well-defined for the problem given in (2.1), a direct comparison of the rates for the two methods is not really possible. Our result is a natural extension of the standard, best-known worst-case complexity of any first-order method for general non-convex optimization [57].

Our result does not yet provide a termination criterion that would guarantee that $\|f(\bar{x})\| \leq \epsilon$, where $\bar{x}$ is the last iterate returned by the algorithm. However, we believe the analysis provides a foundation for establishing such a criterion. In particular, while we simply bound the expected iteration complexity in this chapter, bounding the tail of the complexity distribution ought to follow from the analysis here.

3.2 STORM as a Random Process

We recall Algorithm 1 from Chapter 2. Algorithm 1 generates a random process; the sources of randomness are the random models and random estimates constructed on each iteration, based on some random information obtained from the stochastic function $f(x, \xi)$. As before, M_k will denote a random model in the k-th iteration, while we will use the notation $m_k = M_k(\omega)$ for its realizations. As a consequence of using random models, the iterates X_k, the trust-region radii Δ_k and the steps S_k are also random quantities, and so $x_k = X_k(\omega)$, $\delta_k = \Delta_k(\omega)$, $s_k = S_k(\omega)$ will denote their respective realizations. Similarly, let random quantities $\{F_k^0, F_k^s\}$ denote the estimates of $f(X_k)$ and $f(X_k + S_k)$, with their realizations denoted by $f_k^0 = F_k^0(\omega)$ and $f_k^s = F_k^s(\omega)$. In other words, Algorithm 1 results in a stochastic process $\{M_k, X_k, S_k, \Delta_k, F_k^0, F_k^s\}$. Our goal is to show that under essentially the same assumptions on the sequences $\{M_k\}$ and $\{F_k^0, F_k^s\}$ as in Chapter 2, the resulting stochastic process has a desirable convergence rate.

The key to the convergence analysis given in Section 2.5, and hence its extension here, lies in the assumption that the accuracy (but not necessarily the probability with which the accuracy holds) improves in coordination with the perceived progress of the algorithm. The main challenge of the convergence rate analysis that we will perform in this chapter lies in the fact that, while in the deterministic case (see [37] for a convergence rate analysis of trust-region methods for deterministic functions utilizing random models), the function $f(x)$ never increases from one iteration to another, this can easily happen in the stochastic case. The analysis is based on properties of supermartingales where the increments of a supermartingale depend on the change in the true function value between iterates (which as we will show, *tend* to decrease). To make the analysis simpler, we need a technical assumption that these increments are bounded from above. Hence, we summarize our assumptions on f in Assumption 3.2.1:

Assumption 3.2.1. *We assume that all iterates x_k generated by Algorithm 1 satisfy $x_k \in \mathcal{X}$, where $\mathcal{X}$ is an open bounded set in $\mathbb{R}^n$. We also assume that f and its gradient*

∇f *are L-Lipschitz continuous for all* $x \in \mathcal{X}$ *and that*

$$0 \leq f(x) \leq F_{\max} \quad \forall x \in \mathcal{X}$$

The assumptions of Lipschitz continuity and boundedness of f from below and above in any bounded set are standard, and were in fact cast in the previous chapter, see 2.5.1. Here for simplicity and without loss of generality, we assume in 3.2.1 that the lower bound on f is nonnegative. The additional assumption in 3.2.1 that x_k remains in a bounded set is necessary, to ensure an upper bound on $f(x_k)$. In the deterministic case, the iterates remain in the level set $f(x) \leq f(x^0)$, which is often assumed to be bounded; in the stochastic case, keeping iterates in a bounded set can be enforced by minor modifications to Algorithm 1 or, alternatively, shown to hold with overwhelming probability for a sufficiently large $\mathcal{X}$ containing $f(x^0)$.

3.2.1 Assumptions on the algorithm

We remind the reader of our previous definitions of α-probabilistically κ-fully linear models and β-probabilistically ϵ_F-accurate estimates given in Definitions 2.4.4 and 2.4.5, respectively. Throughout this chapter, we will use the same notation used in those definitions.

Having recalled these definitions, we now state some assumptions that will be necessary in our convergence rate analysis.

Assumption 3.2.2. *The following hold for the quantities used in Algorithm 1*

1. *The model Hessians* $\|H_k\|_2 \leq \kappa_{bhm}$ *for some* $\kappa_{bhm} \geq 1$*, for all* k*, deterministically.*
2. *The constant* η_2 *is chosen to satisfy* $\eta_2 \geq \max\{\kappa_{bhm}, \frac{6\kappa_{ef}}{\kappa_{fcd}}\}$.
3. *The sequence of random models* M_k *generated by Algorithm 1 is* α*-probabilistically* κ*-fully linear, for some* $\kappa = (\kappa_{ef}, \kappa_{eg})$ *and for a sufficiently large* $\alpha \in (0, 1)$.
4. *The sequence of random estimates* $\{F_k^0, F_k^s\}$ *generated by Algorithm 1 is* β*-probabilistically* ϵ_F*-accurate for* $\epsilon_F \leq \frac{1}{4}\eta_1\eta_2$ *and for a sufficiently large* $\beta \in (0, 1)$.

We remark that in Assumption 3.2.2, (1.) is simply a restatement of Assumption 2.5.2, (2.) is immediately implied by (2.21) (the constant given here simply works more transparently in the subsequent analysis), (3.) is unchanged from Assumption 2.5.3, and (4.) is virtually unchanged from Assumption 2.5.3, while the condition on ϵ_F is immediately implied by (2.22).

Let us define additional random indicator variables $G_k = \mathbf{1}\{I_k \cap J_k\}$ and $B_k = 1 - \mathbf{1}\{I_k \cup J_k\}$. Hence, $G_k = 1$ on iterations for which both the models and the estimates are sufficiently accurate and $B_k = 1$ when neither are sufficiently accurate. Due to Assumption 3.2.2,

$$P\{G_k = 1|\mathcal{F}_{k-1}^{M \cdot F}\} \geq \alpha\beta \text{ and } P\{B_k = 1|\mathcal{F}_{k-1}^{M \cdot F}\} \leq (1-\alpha)(1-\beta).$$

Choosing constants To further simplify expressions for various constants we will assume that $\eta_1 = 0.1$ and $\kappa_{fcd} = 0.5$, both of which are typical values for these constants. This will imply the choice $\eta_2 = 12\kappa_{ef}$ and $\epsilon_F = \frac{3}{10}\kappa_{ef}$ to satisfy Assumption 3.2.2. We will also assume that $\gamma < 2$, which is, again, standard. To simplify expressions further we will consider $\kappa_{ef}, \kappa_{eg} \geq 20$. It is clear that if either κ_{ef} or κ_{eg} happens to be smaller, somewhat better bounds than the ones we derive here will result. We are interested in deriving bounds for the case when κ_{eg} may be large. The analysis can be performed for any other values of the above constants, hence the choice here is done merely for convenience and simplicity of presentation.

We now seek to bound the expected number of steps that Algorithm 1 takes until $\|\nabla f(X_k)\| \leq \epsilon$ occurs. We will cite various intermediate results from Section 2.5 that led to Theorem 2.5.3.

3.3 Stopping time of a stochastic process

We consider a random process

$$\Phi_k = \nu f(x_k) + (1-\nu)\Delta_k^2$$

where $\nu \in (0,1)$ is a deterministic, large enough constant, which we will define later. Clearly $\Phi_k \geq 0$. Denote realizations of Φ_k by ϕ_k. Let $V_k = \Phi_{k+1} - \Phi_k$.

Given a stochastic process $\{X_k\}$, we say T is a *stopping time for* $\{X_k\}$ provided that $P(T < \infty) = 1$ and the random event $\{T = k\}$ is determined conditioned on $X_1, \ldots, X_k$.

Define a random time

$$T_\epsilon = \inf\{k \geq 0 : \|\nabla f(X_k)\| \leq \epsilon\}.$$

The following is an immediate consequence of Theorem 2.5.3:

Theorem 3.3.1. *T_ϵ is a stopping time for the stochastic process X_k defined by Algorithm 1.*

As stated, our goal is to bound the expected stopping time $\mathbb{E}(T_\epsilon)$. Towards this end, define

$$\Delta_\epsilon = \frac{\epsilon}{\zeta} \quad \text{and} \quad \zeta = \kappa_{eg} + \max\left\{\eta_2, \frac{8\kappa_{ef}}{\kappa_{fcd}(1-\eta_1)}\right\}, \tag{3.1}$$

which with our choice of algorithmic parameters implies $\zeta = \kappa_{eg} + 160\kappa_{ef}$.

Conditioned on $\|\nabla f(X_k)\| \geq \epsilon$, or equivalently, conditioned on $T_\epsilon > k$, whenever $\Delta_k \leq \Delta_\epsilon$, we have

$$\zeta\Delta_k \leq \|\nabla f(X_k)\|. \tag{3.2}$$

For simplicity and without loss of generality, we assume that there exist integers i, j with $i, j > 0$ such that $\Delta_0 = \gamma^i\Delta_\epsilon$ and $\Delta_{\max} = \gamma^j\Delta_\epsilon$.

From the analysis in Chapter 2, we have the following two results.

Lemma 3.3.1. **($\Delta_k \leq \Delta_\epsilon$)** *Let Assumptions 3.2.1 and 3.2.2 hold. Let α and β satisfy*

$$\frac{\alpha\beta}{(1-\alpha)(1-\beta)} \geq \max\left\{\frac{1}{2}, 2 + \frac{(2\kappa_{eg} + 320\kappa_{ef} + 3L)16\kappa_{eg}}{\kappa_{eg} + 160\kappa_{ef}}\right\}$$

and

$$\nu = \max\left\{\frac{1}{2}, 1 - \frac{\kappa_{eg} + 160\kappa_{ef}}{8(\gamma^2 - 1/\gamma^2)\kappa_{eg} + \kappa_{eg} + 160\kappa_{ef}}\right\}. \tag{3.3}$$

Conditioned on $T_\epsilon > k$, whenever $\Delta_k \leq \Delta_\epsilon$, we have

$$\begin{aligned} \Delta_{k+1} &\geq \min(\Delta_{\max}, \gamma\Delta_k)G_k + \gamma^{-1}\Delta_k(1 - G_k) \\ \mathbb{E}[V_k|\mathcal{F}_{k-1}^{M\cdot F}] &\leq -\Theta_1\|\nabla f(X_k)\|\Delta_k \leq -\Theta_1\epsilon\Delta_k, \end{aligned} \tag{3.4}$$

where $\Theta_1 = \frac{1}{64\kappa_{eg}}$ (under the assumption that $\kappa_{eg} \geq 20$).

Proof. The proof is an immediate consequence of the analysis of Case 1 ($\|\nabla f(X_k)\| \geq \zeta\Delta_k$) in Theorem 2.5.2 and Corollary 2.5.1, due to the definition of Δ_ϵ and (3.2). The only difference is in the particular choice of constants given in Assumption 3.2.2. The dynamics given in (3.4) follow from the definitions of G_k and V_k. □

To quickly recall the main idea of Case 1 in Theorem 2.5.2, the models and estimates are accurate at iteration k in this Case, and so a successful step is guaranteed; hence, $\Delta_{k+1} = \min(\Delta_{\max}, \gamma\Delta_k)$ and $f(X_k) - f(X_{k+1})$ is bounded from below by a constant multiple (here, Θ_1) of $\Delta_k\|\nabla f(X_k)\| \geq \Delta_k\epsilon$. On the other hand, if $B_k = 1$, i.e., the models and estimates are both inaccurate, then in the worst case, the algorithm accepts a bad step. In this worst case, $f(X_k) - f(X_{k+1})$ may be negative, but it is at least bounded from below by a constant multiple of $-\Delta_k\|\nabla f(X_k)\|$. As for the trust region parameter, Δ_{k+1} will increase if the false step is taken, but may decrease if it rejected; hence, $\Delta_{k+1} \geq \gamma^{-1}\Delta_k$. Finally if exactly one of the model or estimates are inaccurate, then in the worst case, the step will be rejected, resulting in $f(X_k) - f(X_{k+1}) \geq 0$ and $\Delta_{k+1} = \gamma^{-1}\Delta_k$.

Lemma 3.3.2. **($\Delta_k > \Delta_\epsilon$)** *In addition to the assumptions of Lemma 3.3.1, let*

$$\begin{aligned} & (1-\alpha)(1-\beta) \\ \leq\ & \min\left\{\tfrac{2\kappa_{eg}+160\kappa_{ef}}{10\kappa_{eg}+1600\kappa_{ef}+16\kappa_{eg}(2\kappa_{eg}+320\kappa_{ef}+3L)}, \tfrac{1}{2+2\kappa_{eg}+320\kappa_{ef}+6L}\right\}. \end{aligned}$$

Conditioned on $T_\epsilon > k$, whenever $\Delta_k > \Delta_\epsilon$, we have

$$\begin{aligned} \Delta_{k+1} &\geq \Delta_\epsilon \\ \mathbb{E}[V_k|\mathcal{F}_{k-1}^{M\cdot F}] &\leq -\Theta_2\epsilon\Delta_\epsilon, \end{aligned} \tag{3.5}$$

for $\Theta_2 = \min\left\{\frac{\gamma^2-1}{(2\kappa_{eg}+320\kappa_{ef})\gamma^2}, \frac{1}{80\kappa_{eg}}\right\}$.

Proof. As in the proof of Lemma 3.3.1, except the result is a consequence of Case 2 ($\|\nabla f(X_k)\| < \zeta\Delta_k$) of Theorem 2.5.2. □

Similarly, we quickly recall the main idea of Case 2 in Theorem 2.5.2. If $B_k = 0$, then in the worst case $f(X_k) - f(X_{k+1}) \geq 0$, $\Delta_{k+1} = \gamma\Delta_k$, and thus $V_k = \Phi_{k+1} - \Phi_k$ is bounded from above by a constant multiple (here, Θ_2) of $-\Delta_k^2 \leq -\epsilon\Delta_\epsilon$. On the other hand, if $B_k = 1$, then a bad step may be taken and $f(X_k) - f(X_{k+1})$ may be negative, but is once again bounded from below by a constant multiple (again, Θ_2) of $-\Delta_k\|\nabla f(X_k)\| \leq -\epsilon\Delta_\epsilon$.

Based on the stochastic process $\{V_k, \Delta_k\}$ described by (3.4) and (3.5), we consider a renewal process, where renewals are defined by the iterations where $\Delta_k = \Delta_\epsilon$ and consider the sum of V_k obtained between two renewals.

Here we will make use of the assumption in Assumption 3.2.1 that there exists a constant $F_{\max}$ such that $f(X_k) \leq F_{\max}$ at every iteration k. The immediate consequence of this assumption is that the V_k process has bounded increments. In particular, we get that for any k,

$$|V_k| = |\Phi_k - \Phi_{k-1}| \leq \nu|(f(X_k) - f(X_{k-1}))| + (1-\nu)|(\Delta_k^2 - \Delta_{k-1}^2)| \leq F_{\max} + \Delta_{\max}^2. \quad (3.6)$$

To summarize the behavior of the Δ_k process from Lemmas 3.3.1 and 3.3.2, we introduce an auxiliary W_k birth-death process defined by

$$\begin{aligned} P(W_k = 1|\mathcal{F}_{k-1}^{M\cdot F}) &= p \qquad \equiv \alpha\beta \\ P(W_k = -1|\mathcal{F}_{k-1}^{M\cdot F}) &= 1 - p. \end{aligned} \quad (3.7)$$

Then we can write

$$\Delta_{k+1} \geq \min(\Delta_k e^{\lambda W_k}, \Delta_\epsilon), \quad (3.8)$$

where $\lambda = \ln(\gamma)$. Observing that $\Theta_2 \leq \Theta_1$, by Lemmas 3.3.1 and 3.3.2 we have

$$\mathbb{E}(V_k|\mathcal{F}_{k-1}^{M\cdot F}, T_\epsilon > k) \leq -\Theta_2\epsilon\Delta_k \quad (3.9)$$

or, equivalently,

$$\mathbb{E}(\Phi_k|\mathcal{F}_{k-1}^{M\cdot F}, T_\epsilon > k) \leq \Phi_{k-1} - \Theta_2\epsilon\Delta_k \tag{3.10}$$

from which we conclude that, conditioned on $T_\epsilon > k$, Φ_k is a supermartingale.

We define the renewal process $\{A_n\}$ as follows: $A_0 = 0$ and
$A_n = \inf\{m > A_{n-1} : \Delta_m \geq \Delta_\epsilon\}$. Naturally, the interarrival times of this renewal process are given for all $k \geq 1$ by

$$\tau_n = A_n - A_{n-1},$$

As a final piece of notation, we define the counting process

$$N(k) = \max\{n : A_n \leq k\},$$

which is the number of renewals that occur before time k.

First, we have a lemma which relies on the simple structure of the process $\{W_k\}$ to bound $\mathbb{E}[\tau_n]$.

Lemma 3.3.3. *Let the random process Δ_k satisfy* (3.7) *and* (3.8) *and let τ_n be defined as above. Then, for all n*

$$\mathbb{E}[\tau_n] \leq p/(2p-1)$$

Proof. We have

$$\begin{aligned}\mathbb{E}[\tau_n] = \;& \mathbb{E}[\tau_n|\Delta_{A_{n-1}} > \Delta_\epsilon]P\{\Delta_{A_{n-1}} > \Delta_\epsilon\} + \mathbb{E}[\tau_n|\Delta_{A_{n-1}} = \Delta_\epsilon]P\{\Delta_{A_{n-1}} = \Delta_\epsilon\} &(3.11)\\ \leq \;& \max\{\mathbb{E}[\tau_n|\Delta_{A_{n-1}} > \Delta_\epsilon], \mathbb{E}[\tau_n|\Delta_{A_{n-1}} = \Delta_\epsilon]\} &(3.12)\end{aligned}$$

By (3.8), we trivially have

$$\mathbb{E}[\tau_n|\Delta_{A_{k-1}} > \Delta_\epsilon] = 1. \tag{3.13}$$

Bounding the other term in (3.11) is less trivial, but is still simple, after observing that conditioned on $\Delta_{A_{n-1}} = \Delta_\epsilon$, the process $\{\Delta_{A_{n-1}}, \Delta_{A_{n-1}+1}, \ldots, \Delta_{A_n}\}$ is a geometric random walk between two returns to the same state (i.e. Δ_ϵ). The expected number of steps between such two returns is well known and can be easily estimated via properties of

ergodic Markov chains. In particular, under the assumption $p > 1/2$ we have

$$\mathbb{E}[\tau_n | \Delta_{A_{k-1}} > \Delta_\epsilon] \leq p/(2p-1). \tag{3.14}$$

Substituting (3.13) and (3.14) into (3.11) completes the proof. □

We now bound the number of renewals that can occur before time T_ϵ.

Lemma 3.3.4.

$$\mathbb{E}[N(T_\epsilon)] \leq \frac{\Phi_0}{\Theta_2 \epsilon \Delta_\epsilon} + \frac{\Delta_0}{\Delta_\epsilon}.$$

Proof. For ease of notation, let $k \wedge T_\epsilon = \min\{k, T_\epsilon\}$. Consider the sequence of random variables

$$R_{k \wedge T_\epsilon} = \Phi_{k \wedge T_\epsilon} + \Theta_2 \epsilon \sum_{j=0}^{k \wedge T_\epsilon} \Delta_j,$$

where Θ_2 is as in (3.10). Observe that $R_{k \wedge T_\epsilon}$ is a supermartingale with respect to $\mathcal{F}_{k-1}^{M \cdot F}$. Indeed, by (3.10),

$$\begin{aligned}
\mathbb{E}[R_{k \wedge T_\epsilon} | \mathcal{F}_{k-1}^{M \cdot F}] &= \mathbb{E}[\Phi_{k \wedge T_\epsilon} | \mathcal{F}_{k-1}^{M \cdot F}] + \mathbb{E}\left[\Theta_2 \epsilon \sum_{j=0}^{k \wedge T_\epsilon} \Delta_j | \mathcal{F}_{k-1}^{M \cdot F} \right] \\
&\leq \Phi_{k-1} - \Theta_2 \epsilon \Delta_k + \Theta_2 \epsilon \sum_{j=0}^{k \wedge T_\epsilon} \Delta_j \\
&= \Phi_{k-1} + \Theta_2 \epsilon \sum_{j=0}^{(k-1) \wedge T_\epsilon} \Delta_j = R_{(k-1) \wedge T_\epsilon}.
\end{aligned}$$

Moreover, for all $k \geq T_\epsilon$,

$$|R_{k \wedge T_\epsilon}| = |R_{T_\epsilon}| \leq F_{\max} + \Delta_{\max} + \Theta_2 \epsilon \sum_{j=0}^{T_\epsilon} \Delta_j.$$

which implies that $|R_{k \wedge T_\epsilon}|$ is almost surely bounded, since T_ϵ is almost surely bounded. Since T_ϵ is a stopping time, it follows from the Optional Stopping Theorem (see e.g. The-

orem 6.4.1 of [66]) that

$$\Theta_2 \epsilon \mathbb{E}\left[\sum_{j=0}^{T_\epsilon} \Delta_j\right] \leq \mathbb{E}[R_{T_\epsilon}] \leq \mathbb{E}[R_0] = \Phi_0 + \Theta_2 \epsilon \Delta_0. \tag{3.15}$$

By the definition of the counting process $N(k)$, and since the renewal times A_n when $\Delta_k \geq \Delta_\epsilon$ are a subset of the iterations $0, 1, \ldots, T_\epsilon$, we have

$$\Theta_2 \epsilon \sum_{j=0}^{T_\epsilon} \Delta_j \geq \Theta_2 \epsilon N(T_\epsilon) \Delta_\epsilon. \tag{3.16}$$

Inserting (3.16) in (3.15),

$$\mathbb{E}(N(T_\epsilon)) \leq \frac{\Phi_0 + \Theta_2 \epsilon \Delta_0}{\Theta_2 \epsilon \Delta_\epsilon},$$

which concludes the proof. □

We now state the following well-known theorem from stochastic processes literature.

Theorem 3.3.2. ***Wald's Equation (inequality form).*** *Suppose $\{Y_n\}$ is a sequence of independent random variables with $\mathbb{E}[|Y_n|] \leq \mathcal{Y} < \infty$ for all n. If N is a stopping time for the process $\{\sum_{n=0}^{k} Y_n\}$ and if $\mathbb{E}[N] < \infty$, then*

$$\mathbb{E}\left[\sum_{n=1}^{N} Y_n\right] \leq \mathbb{E}[N]\mathcal{Y}.$$

For a proof, see for instance Corollary 6.2.3 of [66]. We now apply this theorem to $A_n = \sum_{i=0}^{n} \tau_i$ and obtain the main result of this chapter.

Theorem 3.3.3. *Let assumptions of Lemma 3.3.1 and 3.3.2 hold. Then*

$$\mathbb{E}[T_\epsilon] \leq \frac{\alpha\beta}{2\alpha\beta - 1}\left(\frac{\Phi_0(\kappa_{eg} + 160\kappa_{ef})}{\Theta_2 \epsilon^2} + \frac{\Delta_0(\kappa_{eg} + 160\kappa_{ef})}{\epsilon} + 1\right).$$

Proof. Since T_ϵ is a stopping time for our main stochastic process, then $N(T_\epsilon) + 1$ is a stopping time for the renewal process $\{A_n : n \geq 0\}$. We know that the interarrival times τ_n are independent, and we know that $\mathbb{E}[|\tau_n|] \leq \frac{p}{2p-1} < \infty$ by Lemma 3.3.3 for all

$n < N_{T_\epsilon+1}$. Therefore, by Wald's equation,

$$\mathbb{E}[A_{N(T_\epsilon)+1}] \leq \frac{p}{2p-1} E[N(T_\epsilon)+1].$$

Since $A_{N(T_\epsilon)+1} \geq T_\epsilon$, we have by Lemmas 3.3.3 and 3.3.4 that

$$\mathbb{E}[T_\epsilon] \leq E[\tau_1] E[N(T_\epsilon)+1] \leq \frac{p}{2p-1} \left(\frac{\Phi_0}{\Theta_2 \epsilon \Delta_\epsilon} + \frac{\Delta_0}{\Delta_\epsilon} + 1 \right).$$

The result is obtained by observing that $\Delta_\epsilon = \epsilon/\zeta$ and substituting the expression in (3.1) for ζ. □

Chapter 4

On α-probabilistically Fully-Linear Models

4.1 Introduction

In this chapter[1], we address the concept of probabilistically fully-linear models defined in Definition 2.4.4, and offer guidance as to practical means to construct such models under some typical assumptions on stochasticity.

It has been noted in DFO literature [8, 21] that, within a DFO trust-region framework, using least-squares regression models often leads to better results than using interpolation models in the presence of noise. In [19, 21] regression models are shown to be fully-linear (fully-quadratic) if the sample set satisfies some geometric conditions. However, this result assumes that the function $f(x)$ being modeled is deterministic. In [48], the authors advocate the use of regression models for stochastic $f(x)$ and show that such models can be made probabilistically fully-linear, relying on results from [19, 21] and on their assumptions on $f(x)$. Aside from that work, there has been no systematic study in the derivative-free or stochastic optimization literature on the quality of local models based on least-squares regression from the point of view of an optimization algorithm. Hence, the goal of this chapter is to improve on results given in [48] and provide bounds on the accuracy of least-

[1]The work in this Chapter is unpublished joint work with Dávid Pál, Francesco Orabona, and Katya Scheinberg.

squares regression models for various choices of sample sets.

We are concerned with constructing a *local* model of a smooth function $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$ centered about a point $y^0 \in \mathbb{R}^n$. However, we assume $f(\cdot)$ cannot be computed exactly, and we only have access to a noisy estimator of f, which we will denote in this chapter by $\tilde{f}$. In particular, we assume

$$\tilde{f}(x) = f(x) + \xi(x), \tag{4.1}$$

where $\xi(x)$ is once again a random variable (whose distribution may or may not depend on x). We will state explicitly our assumptions on ξ in the next section.

The remainder of this chapter is organized as follows: in Section 4.2, we analyze the bound on the error between a local least-squares regression model (centered around $x = 0$, under some assumptions on $f(0)$) and the stochastic function under three different cases of choices of sample sets. In Section 4.3, we summarize the results and compare the bounds and their dependence on problem dimension and sampling rates. In Section 4.4, we provide a simple extension which allows us to build local models around any x (as opposed to $x = 0$). We conclude with Section 4.5, where we present some numerical experiments that illustrate our analysis.

4.2 Local Error Bound for Linear Least-Squares Models

In this section, we will focus on constructing a local linear least-squares approximation of $f(x)$ in $\mathcal{B}(0, \Delta)$ given a set of $p > n$ many points $\{y^1, y^2, \ldots, y^p\} \subset \mathcal{B}(0, \Delta)$ and corresponding noisy function evaluations at those points, $\{\tilde{f}(y^1), \tilde{f}(y^2), \ldots, \tilde{f}(y^p)\}$. We also assume, for now, that $f(0) = 0$, for simplicity of presentation. We will extend our bounds to local models in more general $\mathcal{B}(x, \Delta)$ in Section 4.4 by using simple shifts of the variables and function values.

Using notation

$$Y = \begin{bmatrix} (y^1)^\top \\ (y^2)^\top \\ \vdots \\ (y^p)^\top \end{bmatrix}, \tilde{F} = \begin{bmatrix} \tilde{f}(y^1) \\ \tilde{f}(y^2) \\ \vdots \\ \tilde{f}(y^p) \end{bmatrix},$$

the least-squares problem is defined as

$$\min_{w \in \mathbb{R}^n} \|Yw - \tilde{F}\|, \tag{4.2}$$

and is well-known to admit a closed-form solution $\hat{w} = (Y^\top Y)^{-1} Y^\top \tilde{F}$, provided Y has full column rank.

Due to our interest in fully-linear models, we will evaluate the quality of $\langle \hat{w}, x \rangle$ as an approximation of $f(x)$ in $\mathcal{B}(0, \Delta)$. In particular, we aim to bound the following two quantities:

$$\sup_{x \in \mathcal{B}(0,\Delta)} |f(x) - \langle \hat{w}, x \rangle|, \qquad \sup_{x \in \mathcal{B}(0,\Delta)} \|\nabla f(x) - \hat{w}\|. \tag{4.3}$$

Towards this goal, we will evaluate these two errors incurred by $\hat{w}$ with respect to similar errors of some theoretical linear models, such as the first-order Taylor model, which may not be computable given the stochasticity of $f(x)$. In particular, we will consider a theoretical model w^* of f that admits a "good" absolute error on $\mathcal{B}(0, \Delta)$ (for example, $w^* = \nabla f(0)$). We will denote this theoretical worst-case absolute error as ϵ, i.e.

$$\sup_{x \in \mathcal{B}(0,\Delta)} |f(x) - \langle w^*, x \rangle| \leq \epsilon. \tag{4.4}$$

Of course, if $f(x)$ is smooth and $w^* = \nabla f(0)$, then $\epsilon \in \mathcal{O}(\Delta^2)$. Our goal is to show that, given ϵ, the least-squares solution $\hat{w}$ does not admit a much worse absolute error than w^*, i.e.

$$\sup_{x \in \mathcal{B}(0,\Delta)} |f(x) - \langle \hat{w}, x \rangle| \leq \mathcal{O}(\epsilon). \tag{4.5}$$

Moreover, we want the constants hidden in the big-O notation of $\mathcal{O}(\epsilon)$ to be as small as possible.

We will now define a few additional pieces of notation to be used in this chapter. Let us define a vector of model values associated with the generally unknowable w^*,

$$b^* = Yw^* = \begin{bmatrix} \langle w^*, y^1 \rangle \\ \langle w^*, y^2 \rangle \\ \vdots \\ \langle w^*, y^p \rangle \end{bmatrix}. \tag{4.6}$$

We will suppose that each sample point $y^i, i \in 1, \ldots, p$ can be decomposed into the sum of a deterministic component and a stochastic component

$$\tilde{f}(y^i) = f(y^i) + \eta_i,$$

where η_i is a realization of the random variable $\xi(y^i)$. In vector notation, $\tilde{F} = F + \eta$, where $F = [f(y^1), \ldots, f(y^p)]^\top$ and $\eta = [\eta_1, \ldots, \eta_p]^\top$. We now formally state our assumption on η (and hence on $\xi(x)$):

Assumption 4.2.1. *The random variables η_i are pairwise independent and are each σ-subgaussian, i.e.*

$$\mathbb{E}_{\eta_i}[e^{\lambda \eta_i}] \leq \exp(\sigma^2 \lambda^2 / 2) \quad \forall \lambda \in \mathbb{R}, \forall i \in \{1, \ldots, p\}. \tag{4.7}$$

It is easy to see by taking derivatives of the moment generating function in (4.7) that Assumption 4.2.1 is a sufficient condition for each η_i having mean 0 and bounded variance, i.e. $\mathbb{E}(\eta_i) = 0$ and $\mathbf{Var}(\eta_i) \leq \sigma^2$.

4.2.1 Decomposition Theorem and Initial Bounds

We now state and prove a simple decomposition theorem, which bounds the left hand side in (4.5) by three key error terms:

- the *unavoidable* error incurred by some theoretical linear model, such as a Taylor model, in the sense of (4.4),

- the *approximation* error incurred by using $\hat{w}$ instead of w^* - a term that is independent of the noise, but does depend on the sample set Y , in particular on the smallest eigenvalue of the matrix $Y^\top Y$, and
- the *stochastic* error, which is the direct result of noise and depends on both Y and η.

In the following subsections we will bound these error terms for different selections of Y.

Theorem 4.2.1. *Suppose $y^1, y^2, \ldots, y^p \in \mathcal{B}(0,\Delta)$ have full column rank. Then*

$$\sup_{x\in\mathcal{B}(0,\Delta)} |f(x) - \langle \hat{w}, x\rangle| \leq \epsilon \left(1 + \frac{\Delta\sqrt{p}}{\sqrt{\lambda_{\min}(Y^\top Y)}}\right) + \sup_{x\in\mathcal{B}(0,\Delta)} |x^\top (Y^\top Y)^{-1} Y^\top \eta|. \tag{4.8}$$

Proof. For any $x \in \mathcal{B}(0,\Delta)$,

$$\begin{aligned}
& |f(x) - \langle \hat{w}, x\rangle| \\
\leq\ & |f(x) - \langle w^*, x\rangle| \ + \ |\langle w^*, x\rangle - \langle \hat{w}, x\rangle| \\
=\ & \epsilon \ + \ |x^\top [(Y^\top Y)^{-1}(Y^\top Y)] w^* - x^\top (Y^\top Y)^{-1} Y^\top \tilde{F}| \\
=\ & \epsilon \ + \ |x^\top (Y^\top Y)^{-1} Y^\top b^* - x^\top (Y^\top Y)^{-1} Y^\top (F+\eta)| \\
\leq\ & \epsilon \ + \ \sup_{x\in\mathcal{B}(0,\Delta)} |x^\top (Y^\top Y)^{-1} Y^\top (b^* - F)| + \sup_{x\in\mathcal{B}(0,\Delta)} |x^\top (Y^\top Y)^{-1} Y^\top \eta|
\end{aligned}$$

We bound the middle term as

$$\begin{aligned}
\sup_{x\in\mathcal{B}(0,\Delta)} |x^\top (Y^\top Y)^{-1} Y^\top (b^* - F)| &\leq \sup_{x\in\mathcal{B}(0,\Delta)} \|x\| \|(Y^\top Y)^{-1} Y^\top\| \|b^* - F\| \\
&\leq \frac{\Delta\epsilon\sqrt{p}}{\sqrt{\lambda_{\min}(Y^\top Y)}},
\end{aligned}$$

where we used in the second inequality the fact that $(Y^\top Y)^{-1} Y^\top$ is the pseudoinverse of Y. □

Analogously, it is simple to derive the following result for the error between the true gradient and the model gradient:

Theorem 4.2.2. *Suppose $y^1, y^2, \ldots, y^p \in \mathcal{B}(0, \Delta)$ have full column rank. Then*

$$\sup_{x \in \mathcal{B}(0,\Delta)} \|\nabla f(x) - \hat{w}\| \leq \|\nabla f(x) - w^*\| + \frac{\epsilon\sqrt{p}}{\sqrt{\lambda_{\min}(Y^\top Y)}} + \|(Y^\top Y)^{-1} Y^\top \eta\|. \tag{4.9}$$

4.2.2 General stochastic error bound

Having provided a decomposition of an error bound into unavoidable, approximation, and stochastic error terms in Theorems 4.2.1 and 4.2.2, we will now provide a general probabilistic large deviation bound specifically on the stochastic error term. That is, we will yield an upper bound on the stochastic error term that holds with probability α and depends on the least eigenvalue of the covariance matrix $Y^\top Y$, the problem dimension n, α itself, and the σ from Assumption 4.2.1, the latter of which can be viewed as a bound on the standard deviation of the noise.

We first need a lemma that can be viewed as a version of Hoeffding's inequality for n-dimensional random variables:

Lemma 4.2.1. *Let a vector of additive noise η satisfy Assumption 4.2.1. Then with probability at least α,*

$$(Y^\top \eta)^\top (Y^\top Y)^{-1} Y^\top \eta < 4\sigma^2 \left(\frac{n \ln(2)}{2} + \ln\left(\frac{1}{1-\alpha}\right)\right).$$

Proof. We will proceed by showing that for all $t \geq 0$,

$$\mathbb{P}_\eta[(Y^\top \eta)^\top (Y^\top Y)^{-1} Y^\top \eta \geq \sigma^2 t] \leq \exp\left(-\frac{t}{4} + \frac{n\ln(2)}{2}\right). \tag{4.10}$$

By Assumption 4.2.1, for any $\lambda \in \mathbb{R}^n$,

$$\begin{aligned} \mathbb{E}[\exp(\langle \lambda, Y^\top \eta \rangle)] &= \mathbb{E}\left[\exp\left(\sum_{i=1}^p \eta_i \langle \lambda, y^i \rangle\right)\right] \\ &= \prod_{i=1}^p \mathbb{E}[\exp(\eta_i \langle \lambda, y^i \rangle)] \leq \prod_{i=1}^p \exp(\sigma^2 (\langle \lambda, y^i \rangle)^2/2). \end{aligned} \tag{4.11}$$

Letting $\Sigma = \sigma^2 Y^\top Y$, (4.11) can be expressed more succinctly as

$$\mathbb{E}[\exp(\langle \lambda, Y^\top \eta \rangle)] \leq \exp\left(\frac{1}{2}\lambda^\top \Sigma \lambda\right). \tag{4.12}$$

for all $\lambda \in \mathbb{R}^n$. We can thus conclude from (4.12) that for all $\lambda \in \mathbb{R}^n$,

$$\mathbb{E}\left[\exp\left(\langle \lambda, Y^\top \eta \rangle - \frac{1}{2}\lambda^\top \Sigma \lambda\right)\right] \leq 1. \tag{4.13}$$

Now suppose $\lambda \sim N(0, \Sigma^{-1})$, i.e. λ is a random variable having a multivariate normal distribution with mean 0 and covariance Σ^{-1}. Suppose further that λ is independent of η. Then, the density of λ is given by

$$d(\lambda) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma^{-1})}} \exp\left(-\frac{1}{2}\lambda^\top \Sigma \lambda\right),$$

and so computing the conditional expectation over λ,

$$\begin{aligned} \mathbb{E}_{\lambda,\eta}\left[\exp\left(\langle \lambda, Y^\top \eta \rangle - \tfrac{1}{2}\lambda^\top \Sigma \lambda\right)\right] = & \int_{\mathbb{R}^n} \mathbb{E}_\eta \left[\exp\left(\langle \lambda, Y^\top \eta \rangle - \frac{1}{2}\lambda^\top \Sigma \lambda\right)\right] d(\lambda) \mathrm{d}\lambda \\ \leq & \int_{\mathbb{R}^n} d(\lambda) \mathrm{d}\lambda = 1, \end{aligned} \tag{4.14}$$

where the inequality comes from (4.13). Manipulating the middle term of (4.14),

$$\begin{aligned} & \int_{\mathbb{R}^n} \mathbb{E}_\eta \left[\exp\left(\langle \lambda, Y^\top \eta \rangle - \frac{1}{2}\lambda^\top \Sigma \lambda\right)\right] d(\lambda) \mathrm{d}\lambda \\ = & \int_{\mathbb{R}^n} \mathbb{E}_\eta \left[\exp\left(\langle \lambda, Y^\top \eta \rangle\right)\right] \exp\left(-\frac{1}{2}\lambda^\top \Sigma \lambda\right) d(\lambda) \mathrm{d}\lambda \\ = & \frac{1}{\sqrt{(2\pi)^n \det(\Sigma^{-1})}} \int_{\mathbb{R}^n} \mathbb{E}_\eta \left[\exp\left(\langle \lambda, Y^\top \eta \rangle\right)\right] \exp\left(-\lambda^\top \Sigma \lambda\right) \mathrm{d}\lambda, \end{aligned}$$

we see that (4.14) can be written as

$$\int_{\mathbb{R}^n} \mathbb{E}_\eta \left[\exp\left(\langle \lambda, Y^\top \eta \rangle - \lambda^\top \Sigma \lambda\right)\right] \mathrm{d}\lambda \leq \sqrt{(2\pi)^n \det(\Sigma^{-1})}. \tag{4.15}$$

Through some straightforward linear algebraic manipulation, the left hand side of (4.15)

can be rewritten so that (4.15) becomes

$$\int_{\mathbb{R}^n} \mathbb{E}_\eta \left[\exp\left(-\frac{1}{2} \left[\left(\lambda - \frac{1}{2}\Sigma^{-1}Y^\top \eta \right)^\top (2\Sigma) \left(\lambda - \frac{1}{2}\Sigma^{-1}Y^\top \eta \right) \right] + \frac{1}{4}(Y^\top \eta)^\top \Sigma^{-1} Y^\top \eta \right) \right] \mathrm{d}\lambda$$
$$\leq \sqrt{(2\pi)^n \det(\Sigma^{-1})}. \tag{4.16}$$

Observing that the density of a multivariate normal distribution with mean $\frac{1}{2}\Sigma^{-1}Y^\top\eta$ and covariance $\frac{1}{2}\Sigma^{-1}$ is

$$d'(\lambda) = \frac{1}{\sqrt{(2\pi)^n \det(\frac{1}{2}\Sigma^{-1})}} \exp\left(-\frac{1}{2} \left[\left(\lambda - \frac{1}{2}\Sigma^{-1}Y^\top \eta \right)^\top (2\Sigma) \left(\lambda - \frac{1}{2}\Sigma^{-1}Y^\top \eta \right) \right] \right),$$

we conclude from (4.16) that

$$\mathbb{E}_\eta \left[\exp\left(\frac{(Y^\top\eta)^\top \Sigma^{-1} Y^\top \eta}{4} \right) \right] \leq \frac{\sqrt{(2\pi)^n \det(\Sigma^{-1})}}{\sqrt{(2\pi)^n \det(\frac{1}{2}\Sigma^{-1})}} \leq 2^{n/2}, \tag{4.17}$$

where we have used properties of the determinant function to obtain the latter inequality. By Markov's inequality, for any $t \geq 0$,

$$\begin{aligned} \mathbb{P}_\eta[(Y^\top\eta)^\top \Sigma^{-1} Y^\top \eta \geq t] &= \mathbb{P}\left[\exp\left(\frac{(Y^\top\eta)^\top \Sigma^{-1} Y^\top \eta}{4} \right) \geq \exp(t/4) \right] \\ &\leq \exp(-t/4)\mathbb{E}_\eta \left[\exp\left(\frac{(Y^\top\eta)^\top \Sigma^{-1} Y^\top \eta}{4} \right) \right] \\ &\leq \exp(-t/4) \cdot 2^{n/2} \\ &= \exp(-t/4 + n\ln(2)/2), \end{aligned} \tag{4.18}$$

where the last inequality is from (4.17). Recalling our notational choice of $\Sigma = \sigma^2 Y^\top Y$, we see that setting the right hand side of (4.18) to $1-\alpha$ and solving for t, we have proven the lemma. □

Using this lemma, it is now simple to yield a probabilistic large deviation bound on the

stochastic noise term.

Theorem 4.2.3. *Let the vector of additive noise η satisfy Assumption 4.2.1. With probability at least α,*

$$\sup_{x\in\mathcal{B}(0,\Delta)} |x^\top (Y^\top Y)^{-1} Y^\top \eta| < 2\sigma\Delta\sqrt{\frac{1}{\lambda_{min}(Y^\top Y)}\left[\frac{n\ln(2)}{2} + \ln\left(\frac{1}{1-\alpha}\right)\right]}.$$

Additionally, with probability at least α,

$$\|(Y^\top Y)^{-1} Y^\top \eta\| < 2\sigma\sqrt{\frac{1}{\lambda_{\min}(Y^\top Y)}\left[\frac{n\ln(2)}{2} + \ln\left(\frac{1}{1-\alpha}\right)\right]}.$$

Proof. We have

$$\begin{aligned}
\sup_{x\in\mathcal{B}(\Delta)} |x^\top (Y^\top Y)^{-1} Y^\top \eta| &= \sup_{x\in\mathcal{B}(\Delta)} |x^\top (Y^\top Y)^{-1/2} (Y^\top Y)^{-1/2} Y^\top \eta| \\
&\leq \sup_{x\in\mathcal{B}(\Delta)} \|x^\top (Y^\top Y)^{-1/2}\| \|(Y^\top Y)^{-1/2} Y^\top \eta\| \\
&= \sup_{x\in\mathcal{B}(\Delta)} \|x^\top (Y^\top Y)^{-1/2}\| \sqrt{(Y^\top \eta)^\top (Y^\top Y)^{-1} Y^\top \eta} \\
&< \Delta\sqrt{\frac{1}{\lambda_{min}(Y^\top Y)}}\left[2\sigma\sqrt{\frac{n\ln(2)}{2} + \ln\left(\frac{1}{1-\alpha}\right)}\right] \quad \text{w.p. } \alpha,
\end{aligned}$$

where the latter inequality is from the magnitude of the eigenvector corresponding to $\lambda_{min}(Y^\top Y)$ and Lemma 4.2.1. The second result follows trivially from $x \in \mathcal{B}(0,\Delta)$. □

Up to this point, all the results we have proven hold for any general sample set Y. In fact, Theorems 4.2.1, 4.2.2, and 4.2.3 are stated in such a way that once one provides a lower bound on $\lambda_{\min}(Y^\top Y)$ for an arbitrary sample set Y, then the upper bounds on the various error terms are immediate. To illustrate this, we will consider three common instances of sample sets Y and explicitly derive the error bounds implied by Theorems 4.2.1 and 4.2.2. In all cases, we will suppose Y is sampled from $\mathcal{B}(0,\Delta)$.

4.2.3 A Baseline Design - Sampling Unitary Matrices

The first form of sampling we propose in this section is the following: let $u_1, u_2, \ldots, u_n$ denote the columns of U, an n-dimensional unitary (i.e. $UU^\top = U^\top U = I_n$) matrix. For

an arbitrary $k \in \mathbb{N}$, we sample (deterministically or randomly) k such unitary matrices $U^1, U^2, \ldots, U^k$, yielding a total of $p = kn$ many samples. We define the jth sample as $y^j = \Delta u_i^l$ where $j = (l-1)n + i$ for $l = 1, 2, \ldots, k$, and $i = 1, 2, \ldots, n$. We shall refer to this as the *identity design*, since it can be seen as a generalization of Monte Carlo sampling along the standard coordinate directions of an identity matrix. If $U^1 = U^2 = \cdots = U^k$, then the least-squares solution to 4.2 is essentially a finite difference gradient, with Δ serving as the width of the finite difference (recall that we assume $f(0) = 0$ for now).

Alternatively, an identity design with $\{U^l\}_{l=1}^k$ generated by random rotations of I_n is considered to be a productive way to generate sample sets for model-based DFO algorithms which use regression models. This idea of random rotations has been successfully used in [48] for stochastic black-box problems.

With any identity design, we observe that $Y^\top Y = k\Delta^2 I_n$, and so $\lambda_{\min}(Y^\top Y) = k\Delta^2 = (p\Delta^2)/n$. Thus, we can immediately conclude the following from Theorems 4.2.1 and 4.2.2:

Theorem 4.2.4. *Let Assumption 4.2.1 hold and suppose Y is a sample of size $p = kn$ sampled in the identity design. Then, with probability at least α,*

$$\sup_{x \in \mathcal{B}(0,\Delta)} |f(x) - \langle \hat{w}, x \rangle| \leq \epsilon(1 + \sqrt{n}) + 2\sigma p^{-1/2} \sqrt{\frac{n^2 \ln(2)}{2} + n \ln\left(\frac{1}{1-\alpha}\right)}.$$

Additionally, with probability at least α,

$$\sup_{x \in \mathcal{B}(0,\Delta)} \|\nabla f(x) - \hat{w}\| \leq \|\nabla f(x) - w^*\| + \frac{\epsilon\sqrt{n}}{\Delta} + \frac{2\sigma}{\Delta\sqrt{p}} \sqrt{\frac{n^2 \ln(2)}{2} + n \ln\left(\frac{1}{1-\alpha}\right)}.$$

Proof. Beginning with the decomposition of error in Theorem 4.2.1,

$$\begin{aligned}
\sup_{x\in\mathcal{B}(0,\Delta)} |f(x) - \langle \hat{w}, x\rangle| &\leq \epsilon\left(1 + \frac{\Delta\sqrt{p}}{\sqrt{\lambda_{\min}(Y^\top Y)}}\right) + \sup_{x\in\mathcal{B}(0,\Delta)} |x^\top (Y^\top Y)^{-1} Y^\top \eta| \\
&\leq \epsilon(1+\sqrt{n}) + \sup_{x\in\mathcal{B}(\Delta)} |x^\top (Y^\top Y)^{-1} Y^\top \eta| \\
&\leq \epsilon(1+\sqrt{n}) + 2\sigma\Delta\sqrt{\frac{1}{\lambda_{min}(Y^\top Y)}\left[\frac{n\ln(2)}{2} + \ln\left(\frac{1}{1-\alpha}\right)\right]} \\
&= \epsilon(1+\sqrt{n}) + 2\sigma\sqrt{\frac{n}{p}\left[\frac{n\ln(2)}{2} + \ln\left(\frac{1}{1-\alpha}\right)\right]} \quad \text{w.p. } \alpha,
\end{aligned}$$

where the third inequality is due to Theorem 4.2.3 and holds with probability α. The first result is immediate and the second is derived analogously using Theorem 4.2.2. □

We remark that Theorem 4.2.4 implies that for any α, even as $p \to \infty$, there is an amount of unavoidable error present; that is, there is an additional amount $\epsilon\sqrt{n}$ over the unavoidable ϵ error incurred by the theoretical model w^* in (4.4). Any remaining error beyond this $\epsilon(1+\sqrt{n})$ is contributed by the stochastic error term, which decays at a rate of $\mathcal{O}(1/\sqrt{p})$.

4.2.4 Sampling from a Uniform Distribution on $\mathcal{B}(0,\Delta)$

We now turn our attention to sampling Y from the uniform distribution on $\mathcal{B}(0,\Delta)$. We will refer to this as the *uniform design* case. As in Section 4.2.3, we have only to obtain a lower bound on $\lambda_{\min}(Y^\top Y)$ and insert this bound into the results from Theorems 4.2.1, 4.2.2, and 4.2.3. Given that Y is now a random matrix with each row being an independent sample from a uniform distribution on $\mathcal{B}(0,\Delta)$, the bound on $\lambda_{\min}(Y^\top Y)$, and hence the overall error bound, will have to hold with some probability. Contrast this with the identity design, in which only the stochastic error bound was probabilistic while the approximation error bound was global. Towards establishing a probabilistic large deviation bound for $\lambda_{\min}(Y^\top Y)$, we state and prove the following lemma.

Lemma 4.2.2. *Suppose $Z \in \mathbb{R}^n$ is a random vector with uniform distribution on $\mathcal{B}(0, \Delta)$. Then,*

$$\mathbb{E}[ZZ^\top] = \frac{\Delta^2}{n+2} I_n.$$

Proof. Consider $\mathbb{E}[Z_i Z_j]$, the expected value of the product of the ith and jth coordinates of Z. When $i \neq j$, $\mathbb{E}[Z_i Z_j] = \mathbb{E}[Z_i \mathbb{E}[Z_j | Z_i]] = \mathbb{E}[Z_i \cdot 0] = 0$.

When $i = j$,

$$\mathbb{E}[Z_i Z_j] = \mathbb{E}[Z_i^2] = \frac{1}{n}\mathbb{E}[\|Z\|^2] = \frac{1}{n}\int_0^\Delta z^2 d_{\|Z\|}(z)\mathrm{d}z, \tag{4.19}$$

where $d_{\|Z\|}(z)$ denotes the density of $\|Z\|$, which has support on $[0, \Delta]$. Recalling that the volume of an n-dimensional ball of radius Δ is

$$V(n, \Delta) = \frac{\pi^{n/2}\Delta^n}{\Gamma(\frac{n}{2}+1)}$$

and defining the surface area of an n-dimensional ball of radius z by

$$A(n, z) = \frac{2\pi^{n/2} z^{n-1}}{\Gamma(\frac{n}{2})},$$

we see that the density in (4.19) can be expressed as

$$\frac{1}{n}\int_0^\Delta z^2 d_{\|Z\|}(z)\mathrm{d}z = \frac{1}{n}\int_0^\Delta z^2 \frac{A(n,z)}{V(n,\Delta)}\mathrm{d}z = \frac{1}{\Delta^n}\int_0^\Delta z^{n+1}\mathrm{d}z = \frac{1}{n+2}\Delta^2.$$

The result follows. □

We state the following concentration inequality from [31] without proof.

Lemma 4.2.3. *Let $Z, y^1, y^2, \ldots, y^p$ be i.i.d. random vectors in $\mathbb{R}^n$. Let λ denote the minimum eigenvalue of $E[ZZ^\top]$ and let $\lambda_{min}(Y^\top Y)$ denote the minimum eigenvalue of $Y^\top Y$. If $p \geq \frac{16n}{\lambda^2}$, then*

$$\mathbb{P}\left[\frac{\lambda_{min}(Y^\top Y)}{p} < \frac{\lambda}{2}\right] \leq \exp\left(-\frac{\lambda^2 p}{8}\right).$$

Equivalently, for any $\alpha \in (0,1)$, if $p \geq \frac{8}{\lambda^2}\max\{2n, \ln(1/(1-\alpha))\}$, then with probability at least α,

$$\lambda_{min}(Y^\top Y) \geq \frac{p\lambda}{2}.$$

The following probabilistic bound on $\lambda_{min}(Y^\top Y)$ follows immediately from Lemmas 4.2.2 and 4.2.3.

Theorem 4.2.5. *Let Y be an i.i.d. sample of size p drawn from a uniform distribution on $\mathcal{B}(\Delta)$. For any $\alpha \in (0,1)$, if $p \geq 16(n+2)^2\Delta^{-4}\max\{2n, \ln(1/(1-\alpha))\}$, then*

$$\lambda_{min}(Y^\top Y) \geq \frac{p\Delta^2}{2(n+2)}$$

holds with probability at least α.

Theorem 4.2.5 gives us the desired probabilistic lower bound on $\lambda_{\min}(Y^\top Y)$. Thus, we have the following result, the proof of which is analogous to the one given for Theorem 4.2.4.

Theorem 4.2.6. *Let $\alpha \in (0,1)$. Let Assumption 4.2.1 hold with constant σ and suppose Y is an i.i.d. sample of size $p \geq 16(n+2)^2\Delta^{-4}\max\{2n, \ln(1/(1-\alpha))\}$ drawn from a uniform distribution on $\mathcal{B}(0,\Delta)$. Then, with probability at least α, the following bound holds:*

$$\sup_{x\in\mathcal{B}(0,\Delta)} |f(x) - \langle \hat{w}, x\rangle| \leq \epsilon(1+\sqrt{2(n+2)}) + \frac{2\sigma}{\sqrt{p}}\sqrt{2(n+2)\left[\frac{n\ln(2)}{2} + \ln\left(\frac{1}{1-\alpha}\right)\right]}.$$

Additionally, with probability at least α,

$$\sup_{x\in\mathcal{B}(0,\Delta)} \|\nabla f(x) - \hat{w}\| \leq \|\nabla f(x) - w^*\| + \frac{\epsilon\sqrt{2(n+2)}}{\Delta} + \frac{2\sigma}{\Delta\sqrt{p}}\sqrt{2(n+2)\left[\frac{n\ln(2)}{2} + \ln\left(\frac{1}{1-\alpha}\right)\right]}.$$

Notice that, as in the identity design case, we have established a bound on the approximation error which is in $\mathcal{O}(\epsilon\sqrt{n})$. The obvious difference is that the bound in this uniform design case can only hold probabilistically, while the bound in the identity case was deterministic. Once again, we observe an unavoidable amount of error in $\mathcal{O}(\epsilon\sqrt{n})$ that results from committing to linear regression models, while the stochastic error once again can be made arbitrarily small by increasing the sample size, decaying at a rate in $\mathcal{O}(1/\sqrt{p})$.

4.2.5 Strongly Λ-poised sample sets

In the DFO literature, it has become increasingly commonplace to measure the quality of a design matrix Y in terms of something known as Λ-poisedness, or, in the case of least-squares regression, strong Λ-poisedness. In [21], it has been shown that if a model-based algorithm such as Algorithm 1 uses regression models built on strongly Λ-poised sample sets, and there is no stochastic noise, then the algorithm converges to a first (or second) order stationary point. In [8], the authors rely on strongly Λ-poised sample sets to construct models of a stochastic function $f(x)$ to yield a convergent algorithm. Essentially, Λ-poisedness can be used to bound the quantity $\lambda_{\min}(Y^\top Y)$; using this bound, the authors of [8] (under some conditions) are able to derive probabilistically fully-linear models, which ensures convergence. The actual sample sets that they propose using in numerical experiments are composed of multiple randomly rotated identity matrices, which is what we called the identity design case in this chapter, but are also properly strongly 1-poised sets. Hence, our results for the identity design case will apply to their analysis, but as we will show here, these bounds are better than the general bounds that hold for strongly Λ-poised sets. In [68], the authors' convergence theory depends critically on the choice of interpolation (as opposed to regression) set on a given iteration being Λ-poised. In that work, they maintain a Λ-poised set of $n+1$ (or more) design points and use Monte Carlo sampling at those design points. This can also be viewed as a strongly Λ-poised sample set with replicated sample points. The goal of this section is to show that we can apply the same framework of analysis as in the previous two design cases to compute error bounds for regression models constructed on strongly Λ-poised sample sets. However, we will show that the bounds we obtain have a strictly worse dependence on n, because we are not exploiting any special properties of these general sets as we did in the identity design and uniform design cases.

In the case of linear models, which is the focus of this chapter, we have the following specialized definition of strong Λ-poisedness:

Definition 4.2.1. *A set of points* $\{y^1, \ldots, y^p\} \in \mathcal{B}(0, \Delta)$ *is* strongly Λ-poised for regression

in $\mathcal{B}(0,\Delta)$ *provided for all* $x \in \mathcal{B}(0,\Delta)$*, there exists some* $\lambda(x) \in \mathbb{R}^p$ *satisfying*

$$Y^\top \lambda(x) = x, \qquad \|\lambda(x)\| \leq \frac{n+1}{\sqrt{p}}\Lambda. \tag{4.20}$$

Observe that Λ is always greater than or equal to 1. In [21], Definition 4.2.1 is stated for the scaled case of $\Delta = 1$; in this chapter, we chose to have the radius Δ be explicitly present. Thus, the definition presented here is slightly different, but is equivalent in the case of linear models. This choice in presentation makes direct comparisons to the other two designs discussed so far immediately clear. We now demonstrate through Lemma 4.2.4 that the condition number $\lambda_{\min}(Y^\top Y)$ that we have used in our bounds thus far can be directly related to the concept of strong Λ-poisedness.

Consider the reduced singular value decomposition $Y = \hat{U}\hat{\Sigma}\hat{V}^\top$, so that $\hat{U} \in \mathbb{R}^{p\times n}, \hat{\Sigma} \in \mathbb{R}^{n\times n}$, and $\hat{V} \in \mathbb{R}^{n\times n}$. We prove the following bound that we need on

$$\frac{1}{\lambda_{\min}(Y^\top Y)} = \frac{1}{\lambda_{\min}(\hat{V}\hat{\Sigma}\hat{U}^\top\hat{U}\hat{\Sigma}\hat{V}^\top)} = \frac{1}{\lambda_{\min}(\hat{\Sigma}^2)} = \|\hat{\Sigma}^{-1}\|^2,$$

which can be seen as a special case of Theorem 2.8 in [19].

Lemma 4.2.4. *If* Y *is strongly* Λ*-poised for regression in* $\mathcal{B}(0,\Delta)$*, then* $\hat{\Sigma}$ *is nonsingular and satisfies*

$$\|\hat{\Sigma}^{-1}\| \leq \frac{(n+1)\Lambda}{\Delta\sqrt{p}}.$$

Conversely, if $\hat{\Sigma}$ *is nonsingular and* $\|\hat{\Sigma}^{-1}\| \leq \Lambda$*, then* Y *is strongly* Λ*-poised for regression in* $\mathcal{B}(0,\Delta)$*.*

Proof. Let Y be strongly Λ-poised for regression in $\mathcal{B}(0,\Delta)$. Towards contradiction, suppose $\hat{\Sigma}$ is singular. Then, there exists $w' \neq 0$ with $w' \in Ker(Y)$. Thus, for all $x \in Im(Y^\top)$, which is a subspace orthogonal to $Ker(Y)$, we must have $\langle w', x\rangle = 0$. By the definition of Y being Λ-poised for regression in $\mathcal{B}(\Delta)$, we see that for all $x \in \mathcal{B}(0,\Delta)$, $x \in Im(Y^\top)$, as witnessed by some $\lambda(x)$ such that $Y^\top\lambda(x) = x$. So, we in fact have $\langle w', x\rangle = 0$ *for all* $x \in \mathcal{B}(0,\Delta)$, i.e. w' defines a linear function that is constantly zero on all of $\mathcal{B}(\Delta)$. It follows that $w' = 0$, and this is a contradiction.

As for the bound on $\|\hat{\Sigma}^{-1}\|$, observe that

$$\|\hat{\Sigma}^{-1}\| = \|\hat{\Sigma}^{-1}\hat{V}^\top\| = \max_{\|w\|=1} \|\hat{\Sigma}^{-1}\hat{V}^\top w\|, \tag{4.21}$$

where the latter equality follows from the definition of induced matrix norm. Let $\bar{w}$ denote a maximizer of the optimization problem given in (4.21). Then,

$$Y^\top\lambda(x) = x \iff (\hat{U}\hat{\Sigma}\hat{V}^\top)^\top\lambda(x) = x \iff \lambda(x) = \hat{U}\hat{\Sigma}^{-1}\hat{V}^\top x. \tag{4.22}$$

So, by strong Λ-poisedness and (4.22),

$$\|\hat{U}\hat{\Sigma}^{-1}\hat{V}^\top x\| = \|\lambda(x)\| \leq \frac{n+1}{\sqrt{p}}\Lambda. \tag{4.23}$$

We now conclude from (4.21) and (4.23) that

$$\frac{n+1}{\sqrt{p}}\Lambda \geq \max_{x\in\mathcal{B}(\Delta)} \|\hat{U}\hat{\Sigma}^{-1}\hat{V}^\top x\| = \max_{\|w\|=1} \|\hat{\Sigma}^{-1}\hat{V}^\top(\Delta w)\| = \Delta\|\hat{\Sigma}^{-1}\|.$$

If follows that $\|\hat{\Sigma}^{-1}\| \leq \frac{(n+1)\Lambda}{\Delta\sqrt{p}}$, as we meant to show.

As for the converse statement, suppose $\hat{\Sigma}$ is nonsingular and $\|\hat{\Sigma}^{-1}\| \leq \frac{(n+1)\Lambda}{\Delta\sqrt{p}}$. Then, for any $x \in \mathcal{B}(\Delta)$, the minimum-norm solution to the system in (4.20) (which exists by nonsingularity of $\hat{\Sigma}$) satisfies

$$\|\lambda(x)\| \leq \|\hat{U}\hat{\Sigma}^{-1}\hat{V}^\top\|\|x\| \leq \|\hat{\Sigma}^{-1}\|\Delta \leq \frac{(n+1)\Lambda}{\sqrt{p}}.$$

Thus, Y is strongly Λ-poised for regression in $\mathcal{B}(0,\Delta)$. □

From our remarks, the following Theorem is an immediate consequence of Lemma 4.2.4:

Theorem 4.2.7. *Let Y be Λ-poised for regression in $\mathcal{B}(0,\Delta)$. Then,*

$$\frac{1}{\lambda_{\min}(Y^\top Y)} \leq \frac{(n+1)^2\Lambda^2}{p\Delta^2}.$$

Thus, supposing $\Lambda = 1$ so that Y is "perfectly" poised, the bound on $\lambda_{\min}(Y^\top Y)$ is

worse, by a factor of n, than the bounds we have derived for the identity and uniform design case in Sections 4.2.3 and 4.2.4. Recall that the identity design is strongly 1-poised.

Using this bound in Theorem 4.2.7, we can once again easily establish an upper bound on the total error between the regression model and $f(x)$, similarly to Theorems 4.2.4 and 4.2.6.

Theorem 4.2.8. *Let Assumption 4.2.1 hold. Suppose Y is a Λ-poised set in $\mathcal{B}(0,\Delta)$ in the regression sense and Y is of size p. Let $\alpha \in (0,1)$. Then, with probability at least α,*

$$\sup_{x\in\mathcal{B}(0,\Delta)} |f(x) - \langle \hat{w}, x\rangle| < \epsilon\,(1+(n+1)\Lambda) + \frac{2\sigma(n+1)\Lambda}{\sqrt{p}}\sqrt{\frac{n\ln(2)}{2} + \ln\left(\frac{1}{1-\alpha}\right)}.$$

Additionally, with probability at least α,

$$\sup_{x\in\mathcal{B}(0,\Delta)} \|\nabla f(x) - \hat{w}\| \leq \|\nabla f(x) - w^*\| + \frac{\epsilon(n+1)\Lambda}{\Delta} + \frac{2\sigma(n+1)\Lambda}{\Delta\sqrt{p}}\sqrt{\frac{n\ln(2)}{2} + \ln\left(\frac{1}{1-\alpha}\right)}.$$

By Lemma 4.2.4, we see that if $\lambda_{\min}(Y^\top Y)$ is bounded from below, then Y is strongly Λ-poised in the regression sense for some $\Lambda \geq 1$. We will not formalize the following idea here, but this implies that in the uniform case, our probabilistic bound on $\lambda_{\min}(Y^\top Y)$ provided in Theorem 4.2.5 can be interpreted as giving a minimum probability with which a set is well-poised for regression for a suitable Λ.

4.3 Ensuring α-probabilistically κ-fully linear models

Let us summarize the error bounds we derived in Theorems 4.2.4, 4.2.6, and 4.2.8. We observe that each bound can be decomposed into two parts - a *deterministic part* of the form $C_d\epsilon$, where ϵ is as in (4.4), and a *stochastic part* of the form $C_s\sigma$, where σ is the noise parameter defined in Assumption 4.2.1. That is, the bound is generally expressed as

$$\sup_{x\in\mathcal{B}(0,\Delta)} |f(x) - \langle \hat{w}, x\rangle| < C_d\epsilon + C_s\sigma, \qquad \sup_{x\in\mathcal{B}(0,\Delta)} \|\nabla f(x) - \hat{w}\| < C_d\frac{\epsilon}{\Delta} + C_s\frac{\sigma}{\Delta}, \tag{4.24}$$

where the values of C_d and C_s in each case are given in Table 4.3. We define the constant

$$\Omega(n, \alpha) = 2\sqrt{\frac{n \ln(2)}{2} + \ln\left(\frac{1}{1-\alpha}\right)},$$

to clarify presentation, as this function of n and α appears in the stochastic error term for all three bounds that we derived. We remark once again that the dependence on sample size p in C_s is the same for all design cases, as is theoretically expected. We also remark once again that the dependence on n in C_d is strictly worse for the general Λ-poised set design by a factor of $\sqrt{n}$. Again, this is likely due to the fact that in this general case, no particular structure of the sample set could be exploited in the analysis that led to an upper bound, as we were able to do with the identity and uniform design cases.

We now return to the question of constructing an α-probabilistically κ-fully linear model. We will now fix the theoretical model w^* as $\nabla f(0)$, the first-order Taylor model of $f(x)$ centered at 0, and so we naturally obtain

$$\epsilon = \sup_{x \in \mathcal{B}(0,\Delta)} |f(x) - \langle w^*, x \rangle| \leq \kappa_T \Delta^2, \tag{4.25}$$

and

$$\sup_{x \in \mathcal{B}(0,\Delta)} \|\nabla f(x) - w^*\| \leq \kappa_T \Delta, \tag{4.26}$$

where κ_T is the local Lipschitz constant of $\nabla f(x)$ in $\mathcal{B}(0, \Delta)$. We use this upper bound on ϵ in (4.25) and the upper bound on $\|\nabla f(x) - w^*\|$ in (4.26) to prove the following theorem regarding minimum rates at which to sample to obtain an α-probabilistically κ-fully linear model of $f(x)$. Since we assume that n, σ, and κ_T are function-dependent constants, we choose to express the minimum sampling rate $p_{\min}$ as a function of the probability α

	$\mathbf{C_d}$	$\mathbf{C_s}$
Identity design	$1 + \sqrt{n}$	$\Omega(n, \alpha)\sqrt{n}p^{-1/2}$
Uniform design	$1 + \sqrt{2(n+2)}$	$\Omega(n, \alpha)\sqrt{2(n+2)}p^{-1/2}$
Λ-poised set design	$1 + (n+1)\Lambda$	$\Omega(n, \alpha)(n+1)\Lambda p^{-1/2}$

Table 4.1: Values of C_d and C_s in various design cases.

and a trust-region radius Δ. We additionally express $p_{\min}$ as a function of a constant $M > 1$, a multiplicative constant indicating the additional amount of stochastic error we are willing to incur over the unavoidable error and the approximation error. That is, if the deterministic part of the error is $C_d\epsilon$, then we will seek to reduce the stochastic part of the error to $(M-1)C_d\epsilon$.

Theorem 4.3.1. *Let Assumption 4.2.1 hold. Refer to Table 4.3 for values of C_d and $p_{\min}(\alpha, \Delta)$ in each design case. Let $M > 1$, let $\kappa_{ef} = \kappa_{eg} = MC_d\kappa_T$, let $\alpha \in (0, 1)$, and let $\Delta > 0$. If $p \geq p_{\min}(\alpha, \Delta)$ and $\hat{w}$ is built from p samples in the appropriate design case, then with probability α,*

$$\sup_{x \in \mathcal{B}(0,\Delta)} |f(x) - \langle \hat{w}, x \rangle| \leq \kappa_{ef}\Delta^2$$

Additionally, with probability α,

$$\sup_{x \in \mathcal{B}(0,\Delta)} \|\nabla f(x) - \hat{w}\| \leq \kappa_{eg}\Delta$$

Proof. Consider Theorems 4.2.4, 4.2.6, and 4.2.8 for the identity, uniform, and strongly Λ-poised design cases respectively (the results of which were summarized in Table 4.3) and then use both (4.25) and (4.26). The bound $p_{\min}(\alpha, \Delta)$ follows immediately in the identity and strongly Λ-poised design caes. In the uniform case, the second term in the maximum of $p_{\min}(\alpha, \Delta)$ comes from the requirement given in Theorem 4.2.5. □

	$\mathbf{C_d}$	$\mathbf{p}_{\min}(\alpha, \boldsymbol{\Delta})$
Identity design	$1 + \sqrt{n}$	$\dfrac{n\Omega(n,\alpha)^2\sigma^2}{(M-1)^2C_d^2\kappa_T^2\Delta^4}$
Uniform design	$1 + \sqrt{2(n+2)}$	$\max\left\{\dfrac{2(n+2)\Omega(n,\alpha)^2\sigma^2}{(M-1)^2C_d^2\kappa_T^2\Delta^4}, \dfrac{16(n+2)^2\max\{2n, \ln(1/(1-\alpha))\}}{\Delta^4}\right\}$
Λ-poised set design	$1 + (n+1)\Lambda$	$\dfrac{(n+1)^2\Lambda^2\Omega(n,\alpha)^2\sigma^2}{(M-1)^2C_d^2\kappa_T^2\Delta^4}$

Table 4.2: Values of C_d and $p_{\min}(\alpha, \Delta)$ in each design case.

We remark that because the uniform design depends on probabilistic bounds on the $\lambda_{\min}(Y^\top Y)$ term, $p_{\min}(\alpha, \Delta)$ in the uniform design case scales with n in the worst case

as $\mathcal{O}(n^3)$, while in both the identity and Λ-poised set design case, $p_{\min}(\alpha, \Delta)$ scales with $\mathcal{O}(n)$. However, it is worth noting that this $\mathcal{O}(n^3)$ term in the maximum of $p_{\min}(\alpha, \Delta)$ in the uniform design case does not have a dependence on σ, M, or κ_T. Thus, respectively, in situations where σ is large (high levels of noise are present), M is close to 1 (we require the stochastic error term to be quite small), or κ_T is small (the true function f is locally "flat", i.e. it has a small Lipschitz constant in $\mathcal{B}(0, \Delta)$), then the $\mathcal{O}(n^3)$ term will be dominated anyway.

4.4 Constructing Affine Models

So far, we have restricted our analysis to the case where $x \in \mathcal{B}(0, \Delta)$ and $f(0) = 0$ by solving the problem in (4.2), i.e.

$$\min_{w \in \mathbb{R}^n} \|Yw - \tilde{F}\|,$$

where we have assumed each row in Y comes from $\mathcal{B}(0, \Delta)$. However, as motivated in the introduction, we are interested in generating models in a trust region with generally nonzero center points y^0, to be interpreted as the current iterate of a trust region algorithm. Moreover, if $f(y^0) \neq 0$, then we have to consider *affine* models of f in the trust region, i.e. models of the form $w_0 + \langle w, x - y^0 \rangle$. Let

$$\Phi(Y) = \begin{bmatrix} 1 & y_1^0 & \dots & y_n^0 \\ 1 & y_1^1 & \dots & y_n^1 \\ 1 & y_1^2 & \dots & y_n^2 \\ \vdots & \vdots & & \vdots \\ 1 & y_1^p & \dots & y_n^p \end{bmatrix}, \tilde{F}_0 = \begin{bmatrix} \tilde{f}(y^0) \\ \tilde{f}(y^1) \\ \tilde{f}(y^2) \\ \vdots \\ \tilde{f}(y^p) \end{bmatrix},$$

where $y^j = (y_1^j, y_2^j, \dots, y_n^j)$ is an element of $\mathcal{B}(y^0, \Delta)$ for $j = 1, 2, \dots, p$.

We replace the problem in (4.2) with

$$\min_{w \in \mathbb{R}^{n+1}} \|\Phi(Y)w - \tilde{F}_0\|. \tag{4.27}$$

Performing one iteration of Gaussian elimination to the system of equations $\Phi(Y)w = \tilde{F}_0$, one obtains the augmented matrix

$$\left[\begin{array}{cccc|c} 1 & y_1^0 & \dots & y_n^0 & \tilde{f}(y^0) \\ 0 & y_1^1 - y_1^0 & \dots & y_n^1 - y_n^0 & \tilde{f}(y^1) - \tilde{f}(y^0) \\ 0 & y_1^2 - y_1^0 & \dots & y_n^2 - y_n^0 & \tilde{f}(y^2) - \tilde{f}(y^0) \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & y_1^p - y_1^0 & \dots & y_n^p - y_n^0 & \tilde{f}(y^p) - \tilde{f}(y^0) \end{array}\right].$$

Our strategy in the sequel is to choose $w_0 = \tilde{f}(y^0)$ and then solve (in the least-squares sense) the reduced system

$$\left[\begin{array}{ccc|c} y_1^1 - y_1^0 & \dots & y_n^1 - y_n^0 & \tilde{f}(y^1) - \tilde{f}(y^0) \\ y_1^2 - y_1^0 & \dots & y_n^2 - y_n^0 & \tilde{f}(y^2) - \tilde{f}(y^0) \\ \vdots & & \vdots & \vdots \\ y_1^p - y_1^0 & \dots & y_n^p - y_n^0 & \tilde{f}(y^p) - \tilde{f}(y^0) \end{array}\right]. \tag{4.28}$$

Observe that each row in the matrix (4.28) now belongs to $\mathcal{B}(0, \Delta)$ since each $y^j \in \mathcal{B}(y^0, \Delta)$. Thus, it is possible to recover the key results in Theorems 4.2.4, 4.2.6, and 4.2.8. In particular, these theorems were concerned with the use of $\hat{w}$, the least-squares solution to (4.2), to produce a global bound

$$\sup_{x \in \mathcal{B}(0,\Delta)} |f(x) - \langle \hat{w}, x \rangle|.$$

Denote the least squares solution to (4.28) by $\hat{\hat{w}}$. Then, we are now interested in obtaining a global bound on

$$\sup_{x \in \mathcal{B}(y^0,\Delta)} |(f(x) - f(y^0) - (w_0 + \langle \hat{\hat{w}}, x - y^0 \rangle)|. \tag{4.29}$$

By a change of variables $s = x - y^0$, (4.29) can be bounded as

$$\begin{aligned} & \sup_{x \in \mathcal{B}(y^0, \Delta)} |(f(x) - f(y^0)) - (w_0 + \langle \hat{\hat{w}}, x - y^0 \rangle)| \\ = & \sup_{s \in \mathcal{B}(0, \Delta)} |f(y^0 + s) - f(y^0) - (w_0 + \langle \hat{\hat{w}}, s \rangle)| \\ \leq & |f(y^0) - w_0| + \sup_{s \in \mathcal{B}(0, \Delta)} |f(y^0 + s) - \langle \hat{\hat{w}}, s \rangle|. \end{aligned} \tag{4.30}$$

The second summand in the bound given in (4.30) can be immediately bounded using the various results given in Section 4.2. Hence, it is natural to seek an upper bound on the first summand $|f(y^0) - w_0|$ that is at most on the same order as the upper bound on the second. Recall that our overall goal is essentially to obtain a $\mathcal{O}(\Delta^2)$ bound on the approximation of $f(x)$. Thus, we need an estimate w_0 of $f(y^0)$ such that we can bound the *centering error* $|f(y^0) - w_0|$ by a constant times Δ^2. Clearly, this can be achieved by averaging a sufficiently large sample of the unbiased estimator $\tilde{f}(y^0)$. Specifically, due to Chebyshev's theorem and Assumption 4.2.1, we obtain the following lemma.

Lemma 4.4.1. *Let Assumption 4.2.1 hold. Denote $\tilde{f}_p(\cdot) = \sum_{j=1}^{p} \tilde{f}(\cdot, \xi(\cdot))$, a random variable denoting the sum of p realizations of the noisy function value at $\cdot$. Given $\Delta > 0, \kappa_{aff} > 0, \alpha' \in (0,1)$ and*

$$p \geq \frac{\sigma^2}{\kappa_{aff}^2 (1 - \alpha') \Delta^4}, \tag{4.31}$$

then with probability α,

$$|f(\cdot) - \tilde{f}_p(\cdot)| \leq \kappa_{aff} \Delta^2 \tag{4.32}$$

The following corollary of Theorem 4.3.1 and Lemma 4.4.1 summarizes the main result of this chapter.

Corollary 4.4.1. *Let Assumption 4.2.1 hold. Consider a ball $\mathcal{B}(y^0, \Delta)$. Let $\alpha', \alpha'' \in (0,1)$. Let $M > 1$. Suppose a point estimate w_0 of $f(y^0)$ is available satisfying*

$$|f(y^0) - w_0| \leq \kappa_{aff} \Delta^2$$

with probability α'. *For either the identity, uniform, or strongly* Λ*-poised set design, let* C_d *and* $p_{\min}(\alpha'', \Delta)$ *be taken from Table 4.3. Denote* $\hat{w}$ *as the least-squares linear model of* $f(y^0 + s)$ *in* $\mathcal{B}(0, \Delta)$ *built on* $p \geq p_{\min}(\alpha'', \Delta)$ *samples in the appropriate design. Then,* $w_0 + \langle \hat{w}, x - y^0 \rangle$ *is a* $(MC_d\kappa_T + \kappa_{aff}, MC_d\kappa_T)$*-fully linear model of* $f(\cdot)$ *in* $\mathcal{B}(y^0, \Delta)$ *with probability at least* $\alpha' + \alpha'' - 1$.

4.5 A Numerical Experiment

We will briefly illustrate, using a similar experimental setup as in Section 2.7, an empirical preference for using a uniform design in generating regression models in STORM as opposed to an identity design. We reiterate that because the identity design that we chose to analyze in this chapter yields a 1-strongly poised set for regression, there is no need to consider general Λ-strongly poised sets. Using the same benchmark set of 53 unconstrained sum-of-squares problems from Section 2.7.1, we once again consider generating component-wise additive noise $\omega_i \sim \mathcal{U}([-\sigma, \sigma])$ for some parameter $\sigma > 0$. That is, given a function f with m components, the noisy function evaluation $\tilde{f}(x, \omega)$ is once again defined as

$$\tilde{f}(x, \omega) = \sum_{i=1}^{m} (f_i(x) + \omega_i)^2.$$

We will recall Algorithm 7 in the Appendix, but with the following modifications:

- In Line 4 (Regression set update), instead of always uniformly sampling a regression set $Y_k \subset \mathcal{B}(x_k, \delta_k)$, we will use whichever design is appropriate for the experiment.
- In Line 6 (Model building), we will not construct a model Hessian. That is, we will only fit a model gradient g_k through regression as we have throughout this section, and fix $H_k = 0$. The constant f_k in the model building step will be estimated afresh in each iteration as $f_k = \sum_{i=1}^{p_k} \tilde{f}(x_k, \omega_i)$.

We have selected fairly standard constants for a trust-region method in Algorithm 7, namely $\delta_0 = 1, \delta_{\max} = 100, \gamma = 0.5, \eta_1 = 10^{-3}, \eta_2 = 10^{-3}$, and $p_{\min} = n+1$. To yield three different methods, in Line 4 of Algorithm 7, we use as the regression set

1. a uniform design, as in the original statement of Algorithm 7 **(uniform design)**,

2. $\lceil p_k/n \rceil$ copies of a single random unitary matrix **(repeated identity design)**, or

3. $\lceil p_k/n \rceil$ random unitary matrices **(random identity design)**.

In the experiment, for each problem, we set the maximum budget to $1000(n+1)$ function evaluations.

In Figure 4.5, we show data profiles [55] for the experiment. In this experiment, unlike in Section 2.7, when we define the percentage of problems "solved" with respect to $\tau > 0$ in the ratio

$$1 - \tau < \frac{f(x^0) - f'}{f(x^0) - f*},$$

we define f^* as the best true function value found by any of the three different methods within their computational budget. We notice in this experiment that there is a definite preference in terms of speed and robustness for using a uniform design in regression model-building, particularly as the noise becomes more dominant (i.e. $\sigma = 10$).

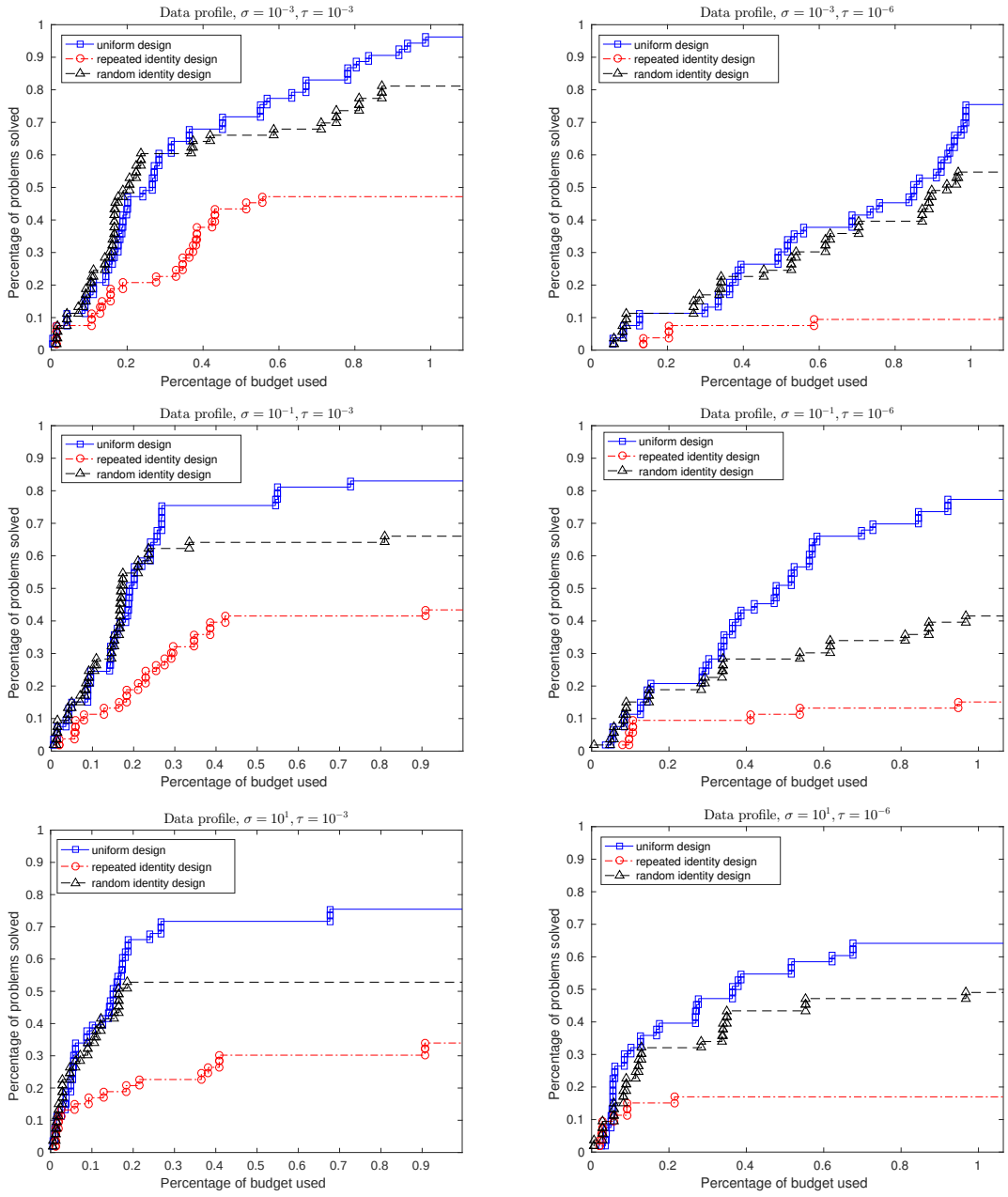

Figure 4.1: Data profiles comparing different regression designs in Algorithm 7.

Chapter 5

Manifold Sampling for ℓ_1 Non-convex Optimization

5.1 Introduction

Temporarily turning away from stochastic problems, this chapter[1] addresses the unconstrained optimization problem $\min\{f(x) : x \in \mathbb{R}^n\}$ when f is deterministic and is of the form

$$f(x) = \sum_{i=1}^{r} |F_i(x)| = \|F(x)\|_1 \tag{5.1}$$

and the function $F : \mathbb{R}^n \to \mathbb{R}^r$ is sufficiently smooth, as formalized in the following assumption.

Assumption 5.1.1. *The function f is of the form* (5.1)*, each F_i is continuously differentiable, and each ∇F_i is Lipschitz continuous with Lipschitz constant L_i; define* $L = \sum_{i=1}^{r} L_i$.

Minimizing the function (5.1) is a special case of more general composite nonsmooth optimization

$$\text{minimize}\,\{f(x) = f_s(x) + h(F(x)) : x \in \mathbb{R}^n\}\,, \tag{5.2}$$

[1]The work in this Chapter is joint work with Jeffrey Larson and Stefan M. Wild. A version of this Chapter has been published in [49].

where f_s and F are smooth but h is nonsmooth with known structure. We find that including f_s and ∇f_s obscures the development of the framework so we do not include them; straightforward modifications can accommodate minimizing $f_s(x) + \|F(x)\|_1$. We focus on the objective function (5.1) in order to succinctly introduce, analyze, and empirically study a general algorithmic framework. Although the form of h studied here is convex, our framework does not require this.

Furthermore, this framework—which we refer to as *manifold sampling*—does not require the availability of the Jacobian ∇F. As a result, manifold sampling is applicable both when inexact values for $\nabla F(x)$ are available and in the derivative-free case, when only the values $F(x)$ are available. In Section 5.2 we motivate the use of the term manifold in the context of functions of the form (5.1) and show that these manifolds can be determined without using Jacobian information. We also review the literature in composite nonsmooth optimization.

Section 5.3 introduces our manifold sampling algorithm. The algorithm uses a smooth model M of the mapping F and proceeds like a traditional trust-region algorithm until it encounters an area of possible nondifferentiability. In the case of the function (5.1), a signal for potential nondifferentiability is directly obtained from the signs of the r component functions at the current iterate. We also propose different mechanisms for incorporating such sign information from the current iterate and nearby points.

Let $M : \mathbb{R}^n \to \mathbb{R}^r$ denote an approximation to F and let $\nabla M : \mathbb{R}^n \to \mathbb{R}^{n \times r}$ denote the Jacobian of M. Under minimal assumptions, in Section 5.4 we show that our algorithm's smooth, local model of the function f,

$$m^f(x^k + s) \approx f(x^k) + \left\langle s, \text{Proj}\left(0, \nabla M\left(x^k\right) \partial h\left(F\left(x^k\right)\right)\right)\right\rangle, \tag{5.3}$$

generates descent directions at the current point x^k. Furthermore, we prove that all cluster points of the algorithm are Clarke stationary points. We show in Section 5.5 that convergence again holds when using sign information from stochastically generated points.

This stochastic sampling proves to be beneficial in practice, as our numerical tests in Section 5.6 demonstrate. Our experiments also underscore the relative robustness of four

variants of the proposed manifold sampling algorithm. The tested deterministic variants include one that performs efficiently when function evaluations occur sequentially as well as variants that can exploit concurrent function evaluations.

5.2 Background

We now introduce notation and provide context for our manifold sampling algorithm. We follow the convention throughout this chapter that finite sets are denoted by capital Roman letters while (possibly) infinite sets are denoted by capital calligraphic letters.

5.2.1 Nonsmooth Optimization Preliminaries

We define the set of points at which a function f is differentiable by $\mathcal{D} \subseteq \mathbb{R}^n$ and its complement by $\mathcal{D}^c$. In smooth optimization, a first-order necessary condition for x to be a local minimum of f is that $\nabla f(x) = 0$. In nonsmooth optimization, if $x \in \mathcal{D}^c$, then one needs a more generalized first-order necessary condition; we achieve this with the generalized gradient ∂f, which is a set-valued function referred to as the *Clarke subdifferential*.

The *Clarke directional derivative at x in the direction d* is given by

$$f^\circ(x; d) = \limsup_{y \to x, t \downarrow 0} \frac{f(y + td) - f(y)}{t}. \tag{5.4}$$

The Clarke subdifferential at x is the set of linear support functions of $f^\circ(x; d)$ when viewed as a function of $d \in \mathbb{R}^n$:

$$\partial f(x) = \{v \in \mathbb{R}^n : f^\circ(x; d) \geq \langle v, d \rangle \text{ for all } d \in \mathbb{R}^n\}. \tag{5.5}$$

By using the definitions (5.4) and (5.5), one can show (see, e.g., [18, Proposition 2.3.2]) that if f is locally Lipschitz near x and attains a local minimum or maximum at x, then $0 \in \partial f(x)$. Thus, $0 \in \partial f(x)$ can be seen as a nonsmooth analogue of the first-order necessary condition $\nabla f(x) = 0$. This condition, $0 \in \partial f(x)$, is referred to as *Clarke stationarity*.

Furthermore, as a consequence of Rademacher's theorem, if f is locally Lipschitz, then

$\mathcal{D}^c$ is a set of Lebesgue measure zero in $\mathbb{R}^n$. In such cases, an equivalent definition (see, e.g., [18, Theorem 2.5.1]) of the Clarke subdifferential is

$$\partial f(x) = \mathbf{co}\left\{ \lim_{y^j \to x} \nabla f(y^j) : \ \{y^j : j \geq 1\} \subset \mathcal{D} \right\}, \tag{5.6}$$

where **co** denotes the convex hull of a set. Equation (5.6) says that $\partial f(x)$ is the convex hull of all limits of gradients of f at differentiable points in an arbitrarily small neighborhood about x.

5.2.2 Manifolds of (5.1)

The signs of the component functions F_i play a critical role in our focus on functions satisfying Assumption 5.1.1. We let *sgn* be the scalar function that returns the values 1, -1, and 0 for positive, negative, and null arguments, respectively, and we define
$\mathbf{sign} : \mathbb{R}^n \to \{-1, 0, 1\}^r$ by

$$\mathbf{sign}\,(x) = \left[\begin{array}{cccc} sgn(F_1(x)) & sgn(F_2(x)) & \cdots & sgn(F_r(x)) \end{array} \right]^\top.$$

We say that $\mathbf{sign}(x)$ returns the *sign pattern* of a point x. There exist 3^r possible sign patterns for any $x \in \mathbb{R}^n$; by indexing these possible sign patterns, we define
$\mathbf{pat}^q \in \{-1, 0, 1\}^r$ for each $q \in \{1, \ldots, 3^r\}$.

For any $x \in \mathbb{R}^n$, its *manifold* $\mathcal{M}(x)$ is the maximal topologically connected set satisfying

$$\mathcal{M}(x) = \{y \in \mathbb{R}^n : \mathbf{sign}(y) = \mathbf{sign}(x)\}.$$

We define the union of all manifolds with the same sign pattern $\mathbf{pat}^q$ as a *manifold set* and denote it by $\mathcal{M}^q = \bigcup_{\{x : \mathbf{sign}(x) = \mathbf{pat}^q\}} \mathcal{M}(x)$. Letting $\mathcal{B}(x, \epsilon) = \{x : \|x\| \leq \epsilon\}$ denote the ball of radius ϵ around a point x, we say that a manifold set $\mathcal{M}^q$ is *active* at x provided that $\mathcal{M}^q \cap \mathcal{B}(x, \epsilon) \neq \emptyset$ for all $\epsilon > 0$.

We note that if Assumption 5.1.1 is satisfied, then the function f is locally Lipschitz. By using chain rule results (dependent on regularity conditions; see, e.g., [18, Definition 2.3.4])

that are ensured by Assumption 5.1.1, the subdifferential of f at a point x is

$$\partial f(x) = \sum_{i:F_i(x)\neq 0} sgn(F_i(x))\nabla F_i(x) + \sum_{i:F_i(x)=0} \mathbf{co}\left\{-\nabla F_i(x), \nabla F_i(x)\right\}, \tag{5.7}$$

where addition is understood to be setwise. Consequently, f is differentiable at x if and only if $F_i(x) = 0$ implies $\nabla F_i(x) = 0$ for $i \in \{1, \ldots, r\}$. This observation motivates our definition of the *nondifferentiable set of* F_i, given by

$$\mathcal{D}_i^c = \{x \in \mathbb{R}^n : F_i(x) = 0 \text{ and } \nabla F_i(x) \neq 0\}. \tag{5.8}$$

Note that for a function satisfying Assumption 5.1.1, $\mathcal{D}^c = \cup_{i=1}^r \mathcal{D}_i^c$. Our algorithmic framework in Section 5.3 is predicated on a relaxation of $\mathcal{D}_i^c$ that does not require the (exact) derivative $\nabla F_i(x)$.

5.2.3 Related Work

The analogue to steepest descent for nonsmooth optimization involves steps along the negative of the minimum-norm element of the subdifferential, $-\operatorname{Proj}\left(0, \partial f(x^k)\right)$. The algorithm that we propose is related to cutting-plane and bundle methods [43], in that the subdifferential in this step is approximated by a finite set of generators.

A class of methods for nonsmooth optimization related to our proposed algorithm is that of gradient sampling. Such methods exploit situations when the underlying nonsmooth function is differentiable almost everywhere by using local gradient information around a current iterate to build a stabilized descent direction [11]. Kiwiel proposes gradient sampling variants in [45], including an approach that performs a line search within a sampling trust region. Such methods use the fact that, under weak regularity conditions (such as those given in Assumption 5.1.1), the closure of the set $\mathbf{co}\left\{\nabla f(y) : y \in \mathcal{B}(x^k, \Delta) \cap \mathcal{D}\right\}$ is a superset of $\partial f(x^k)$. Gradient sampling methods employ a finite sample set $\{y^1, \ldots, y^p\} \subset \mathcal{B}(x^k, \Delta) \cap \mathcal{D}$ and approximate $\partial f(x^k)$ by $\mathbf{co}\{\nabla f(y^1), \ldots, \nabla f(y^p)\}$. The negative of the minimum-norm element of this latter set is used as the search direction.

Constructing this set is more difficult when ∇f is unavailable. A nonderivative version

of the gradient sampling algorithm is shown in [46] to converge almost surely to a first-order stationary point. However, the analysis depends on the use of a Gupal estimate of Steklov averaged gradients as a gradient approximation. Such an approach requires $2n$ function evaluations to compute each approximate gradient. In effect, a single gradient approximation is as expensive to compute as a central-difference gradient approximation, and the approximations must be computed for each point near the current iterate. With this motivation, Hare and Nutini [39] propose an approximate gradient-sampling algorithm that uses standard (e.g., central difference, simplex) gradient approximations to solve finite minimax problems of the composite form minimize $\max_i F_i(x)$. Similar to our focus on the form (5.1), Hare and Nutini [39] model the smooth component functions F_i and use the particular, known structure of their subdifferential $\mathbf{co}\{\nabla F_j(x) : j \in \arg\max_i F_i(x)\}$ within their convergence analysis. Both [39] and [46] employ a line search strategy for globalization. Our method employs a trust-region framework that links the trust-region radius with the norm of a model gradient, which can serve as a stationarity measure.

Other methods also exploit the general structure in composite nonsmooth optimization problems (5.2). When the function h is convex, typical trust-region-based approaches (e.g., [12, 29, 30, 74]) solve the *nonsmooth* subproblem

$$\text{minimize} \left\{ h\left(F(x^k) + \langle s, \nabla M(x^k) \rangle \right) : s \in \mathcal{B}(x^k, \Delta) \right\}. \tag{5.9}$$

The model M is a Taylor expansion of F when derivatives are available. In this case, Griewank et al. [38] propose building piecewise linear models using ∇F from recent function evaluations. These nonsmooth models are then minimized to find future iterates and ∇F information for subsequent models. When derivatives are unavailable, recent work has used sufficiently accurate models of the Jacobian [32, 36]. We follow a similar approach in our approximation of ∇F but employ a fundamentally different subproblem, locally minimizing smooth models related to (5.3). In contrast to (5.9), our subproblem does not rely on the convexity of h.

There also exist derivative-free methods for nonsmooth optimization unrelated to gradient sampling. For example, a generalization of mesh adaptive direct search [2] finds descent

directions for nonsmooth problems by generating an asymptotically dense set of search directions. Similar density requirements exist for general direct search methods [59]. One of the derivative-free methods in [32] employs a smoothing function $f_\mu(x)$ parameterized by a smoothing parameter $\mu > 0$ satisfying

$$\lim_{z \to x, \mu \downarrow 0} f_\mu(z) = f(x),$$

for any $x \in \mathbb{R}^n$. By iteratively driving $\mu \to 0$ within a trust-region framework, the authors prove convergence to Clarke stationary points and provide convergence rates. We note that the Steklov averaged gradients in [46] are also essentially smoothing convolution kernels. Our proposed method requires neither a dense set of search directions nor smoothing parameters.

5.3 Manifold Sampling Algorithm

In order to prove convergence of our algorithm, the models m^{F_i} must sufficiently approximate F_i in a neighborhood of x^k. We require the models to be *fully linear* in the trust region, a notion formalized in the following assumption.

Assumption 5.3.1. *For $i \in \{1, \dots, r\}$, let m^{F_i} denote a twice continuously differentiable model intended to approximate F_i on some $\mathcal{B}(x, \Delta)$. For each $i \in \{1, \dots, r\}$, for all $x \in \mathbb{R}^n$, and for all $\Delta > 0$, there exist constants $\kappa_{i,\text{ef}}$ and $\kappa_{i,\text{eg}}$, independent of x and Δ, so that*

$$\begin{aligned} \left|F_i(x+s) - m^{F_i}(x+s)\right| &\leq \kappa_{i,\text{ef}} \Delta^2 \quad \forall s \in \mathcal{B}(0, \Delta) \\ \left\|\nabla F_i(x+s) - \nabla m^{F_i}(x+s)\right\| &\leq \kappa_{i,\text{eg}} \Delta \quad \forall s \in \mathcal{B}(0, \Delta). \end{aligned}$$

Furthermore, for $i \in \{1, \dots, r\}$ there exists $\kappa_{i,\text{mh}}$ so that $\|\nabla^2 m^{F_i}(x)\| \leq \kappa_{i,\text{mh}}$ for all $x \in \mathbb{R}^n$. For these constants, define $\kappa_\text{f} = \sum_{i=1}^r \kappa_{i,\text{ef}}$, $\kappa_\text{g} = \sum_{i=1}^r \kappa_{i,\text{eg}}$, and $\kappa_\text{mh} = \sum_{i=1}^r \kappa_{i,\text{mh}}$.

Assumption 5.3.1 is nonrestrictive, and one can derive classes of models satisfying the assumption both when ∇F_i is available inexactly and when ∇F_i is unavailable. For instance, in the latter case, the assumption holds when m^{F_i} is a linear model interpolating

F_i at a set of sufficiently affinely independent points (see, e.g., [21, Chapter 10]); in this case, $\kappa_{i,\text{ef}}$ and $\kappa_{i,\text{eg}}$ scale with n and the respective Lipschitz constants of m^{F_i} and ∇m^{F_i}. If ∇F_i is available inexactly, then a model m^{F_i} that uses the inexact gradient while still satisfying Assumption 5.3.1 is a suitable model as well.

5.3.1 Algorithmic Framework

We now outline our algorithm, which samples manifolds (as opposed to gradients or gradient approximations) in order to approximate the subdifferential $\partial f(x^k)$. When x^k is changed, the r component function values $F_i(x^k)$ are computed, immediately yielding $\mathbf{sign}(x^k)$. Then, r component models, m^{F_i}, approximating F_i near x^k are built. Using the value of $\mathbf{sign}(x^k)$, we infer a set of generators, $\mathfrak{G}^k$, using the manifolds that are potentially active at x^k. The set of generators contains information about the manifolds active at (or around) the current iterate; our procedures (Algorithm 3 and Algorithm 4) for constructing $\mathfrak{G}^k$ are detailed in Section 5.3.2. The set $\mathbf{co}\left(\mathfrak{G}^k\right)$ is then used as an approximation to $\partial f(x^k)$.

We let g^k denote the minimum-norm element of $\mathbf{co}\left(\mathfrak{G}^k\right)$,

$$g^k = \text{Proj}\left(0, \mathbf{co}\left(\mathfrak{G}^k\right)\right), \tag{5.10}$$

which can be calculated by solving the quadratic optimization problem

$$\text{minimize}\left\{\frac{1}{2}\lambda^\top (G^k)^\top G^k \lambda : \; e^\top \lambda = 1, \, \lambda \geq 0\right\}, \tag{5.11}$$

where the columns of G^k are the generators in $\mathfrak{G}^k$. A solution λ^* to (5.11) is a set of weights on the subgradient approximations that minimize $\|G\lambda\|^2$. That is, $g^k = G^k\lambda^*$.

Suppose there are $t \leq 3^r$ generators in $\mathfrak{G}^k$ and define the matrix

$$P^k = \begin{bmatrix} | & & | \\ \mathbf{pat}^1 & \cdots & \mathbf{pat}^\top \\ | & & | \end{bmatrix}.$$

Then, since the qth generator in $\mathfrak{G}^k$ (alternatively, the qth column of G^k) is given by $\nabla M(x^k)\mathbf{pat}^q$, we have $G^k = \nabla M(x^k)P^k$. Thus, to maintain the property that the master model gradient is the optimal solution to (5.10), we consider a set of weights $w^k = P^k\lambda^*$ and define a smooth *master model* $m_k^f : \mathbb{R}^n \to \mathbb{R}$,

$$m_k^f(x) = \sum_{i=1}^{r} w_i^k m^{F_i}(x). \tag{5.12}$$

We make a few observations about this choice of w^k. Firstly, as intended by construction,

$$\nabla m_k^f(x^k) = \sum_{i=1}^{r} w_i^k \nabla m^{F_i}(x^k) = \sum_{i=1}^{r} \nabla m^{F_i}(x^k)(P^k\lambda^*)_i = \nabla M(x^k)P^k\lambda^* = G^k\lambda^* = g^k.$$

Secondly, $w_i^k \in [-1, 1]$ for each $i \in \{1, \ldots, t\}$ due to the constraints on λ in (5.11). Thirdly, notice that if G^k contains exactly one generator (i.e., $t = 1$), then $\lambda^* = 1$ is the trivial solution to (5.11) and so the master model in this case is simply

$$m_k^f(x) = \sum_{i=1}^{r} (\mathbf{sign}(x^k))_i m^{F_i}(x) = \langle M(x), \mathbf{sign}(x^k)\rangle.$$

In the kth iteration, the master model will be used in the trust-region subproblem

$$\text{minimize} \left\{ m_k^f(x^k + s) : s \in \mathcal{B}(0, \Delta_k) \right\}. \tag{5.13}$$

Provided the solution $x^k + s^k$ of (5.13) belongs to a manifold that was included in the construction of g^k, we apply a ratio test to determine the successfulness of the proposed step, as in a standard trust-region method. If the manifold containing $x^k + s^k$ is not contributing a generator in $\mathfrak{G}^k$, we augment $\mathfrak{G}^k$ and construct a new g^k. Since there are finitely many manifolds (at most 3^r in the case of (5.1)), this process of adding to $\mathfrak{G}^k$ will terminate.

Our ratio test quantity

$$\rho_k = \frac{\langle F(x^k), \mathbf{sign}(x^k + s^k)\rangle - \langle F(x^k + s^k), \mathbf{sign}(x^k + s^k)\rangle}{\langle M(x^k), \mathbf{sign}(x^k + s^k)\rangle - \langle M(x^k + s^k), \mathbf{sign}(x^k + s^k)\rangle} \tag{5.14}$$

is different from the usual ratio test of actual reduction to predicted reduction in that it considers the function decrease from the perspective of the manifold of the trial step $x^k + s^k$. In particular, our numerator is more conservative than the actual reduction since

$$\begin{aligned}
&\langle F(x^k) - F(x^k + s^k), \mathbf{sign}(x^k + s^k)\rangle - \Big(f(x^k) - f(x^k + s^k)\Big) \\
&= \langle F(x^k) - F(x^k + s^k), \mathbf{sign}(x^k + s^k)\rangle \\
&\quad - \Big(\langle F(x^k), \mathbf{sign}(x^k)\rangle - \langle F(x^k + s^k), \mathbf{sign}(x^k + s^k)\rangle\Big) \\
&= \langle F(x^k), \mathbf{sign}(x^k + s^k) - \mathbf{sign}(x^k)\rangle \leq 0,
\end{aligned}$$

where the inequality holds because $F_i(x^k)[\mathbf{sign}(x^k)]_i = |F_i(x^k)| \geq u_i F_i(x^k)$ for any i and for all $u_i \in [-1, 1]$.

We now state our framework in Algorithm 2, which includes assumptions on algorithmic parameters. We note that an input to the algorithm is a parameter $\kappa_{\mathrm{mh}} \geq 0$ bounding the curvature of the component models. It is always possible to construct fully linear models that are linear (i.e., $\nabla^2 m^{F_i} = 0$); see [73, Algorithm 2 and Theorem 4.5] for an example of how a linear fully linear model can be augmented into a nonlinear fully linear model with bounded curvature. Consequently, the choice of the parameter κ_{mh} does not affect our ability to construct fully linear models.

It is necessary for convergence that the steps s^k achieve a Cauchy-like decrease; however, unlike in a typical trust-region method, this sufficient decrease is needed in the denominator of ρ_k, and not in the step obtained from minimizing (5.13). That is, we require

$$\langle M(x^k) - M(x^k + s^k), \mathbf{sign}(x^k + s^k)\rangle \geq \frac{\kappa_{\mathrm{d}}}{2}\|g^k\| \min\left\{\Delta_k, \frac{\|g^k\|}{\kappa_{\mathrm{mh}}}\right\}. \tag{5.15}$$

Notice that the solution to (5.13) does not necessarily satisfy (5.15). However, a Cauchy-like point given in the following lemma always satisfies (5.15).

Algorithm 2: Manifold sampling.

1 Set parameters $\eta_1 \in (0,1)$, $\kappa_{\text{mh}} \geq 0$, $\frac{1}{\eta_2} \in (\kappa_{\text{mh}}, \infty)$, $\kappa_{\text{d}} \in (0,1)$, $\gamma_{\text{dec}} \in (0,1)$, and $\gamma_{\text{inc}} \geq 1$
2 Choose initial iterate x^0 and trust-region radius $\Delta_0 > 0$; set $k = 0$
3 **while true do**
4 For each F_i, build a model m^{F_i} that is fully linear in $\mathcal{B}(x^k, \Delta_k)$ and satisfies $\sum_{i=1}^{r} \|\nabla^2 m^{F_i}(x)\| \leq \kappa_{\text{mh}}$ for all $x \in \mathbb{R}^n$
5 Build a set of generators $\mathfrak{G}^k$ (from Algorithm 3 or Algorithm 4)
6 **while true do**
7 Build master model m_k^f using the models m^{F_i} and (5.12)
8 **if** $\Delta_k \geq \eta_2 \|\nabla m_k^f(x^k)\|$ **then**
9 **break** (**go to** Line 24)
10 **end**
11 Approximately solve (5.13) to obtain s^k
12 Evaluate $f(x^k + s^k)$
13 **if** $\nabla M(x^k)\mathbf{sign}(x^k + s^k) \in \mathfrak{G}^k$ **then**
14 **if** $x^k + s^k$ *satisfies* (5.15) **then**
15 **break** (**go to** Line 24)
16 **else**
17 $s^k \leftarrow -\kappa_{\text{d}}^{j^*} \Delta_k \frac{\nabla M(x^k)\mathbf{sign}(x^k+s^k)}{\|\nabla M(x^k)\mathbf{sign}(x^k+s^k)\|}$ for j^* defined in (5.16)
18 **go to** Line 12
19 **end**
20 **else**
21 $\mathfrak{G}^k \leftarrow \mathfrak{G}^k \cup \nabla M(x^k)\mathbf{sign}(x^k + s^k)$
22 **end**
23 **end**
24 **if** $\Delta_k < \eta_2 \|\nabla m_k^f(x^k)\|$ (*acceptable* iteration) **then**
25 Update ρ_k through (5.14)
26 **if** $\rho_k > \eta_1$ (*successful* iteration) **then**
27 $x^{k+1} \leftarrow x^k + s^k$, $\Delta_{k+1} \leftarrow \gamma_{\text{inc}} \Delta_k$
28 **else**
29 $x^{k+1} \leftarrow x^k$, $\Delta_{k+1} \leftarrow \gamma_{\text{dec}} \Delta_k$
30 **end**
31 **else**
32 $x^{k+1} \leftarrow x^k$, $\Delta_{k+1} \leftarrow \gamma_{\text{dec}} \Delta_k$
33 **end**
34 $k \leftarrow k + 1$
35 **end**

Lemma 5.3.1. *For any* $\mathbf{pat}_q$ *satisfying* $\nabla M(x^k)\mathbf{pat}_q \in \mathfrak{G}_k$*, if* $M(\cdot)\mathbf{pat}_q$ *is twice continuously differentiable,* $\kappa_{\text{mh}} \geq \max_{s\in\mathcal{B}(0,\Delta_k)} \left\|\nabla^2 M(x^k+s)\mathbf{pat}_q\right\|$*,* $\kappa_{\text{mh}} > 0$*,* $\left\|\nabla M(x^k)\mathbf{pat}_q\right\| > 0$*, and* $\kappa_{\text{d}} \in (0,1)$*, then setting* $s^k = -\kappa_{\text{d}}^{j^*}\Delta_k \frac{\nabla M(x^k)\mathbf{pat}_q}{\left\|\nabla M(x^k)\mathbf{pat}_q\right\|}$*, for*

$$j^* = \max\left\{0, \left\lceil \log_{\kappa_{\text{d}}}\left(\frac{\left\|\nabla M(x^k)\mathbf{pat}_q\right\|}{\Delta_k}\kappa_{\text{mh}}\right)\right\rceil\right\}, \tag{5.16}$$

satisfies $\left\|s^k\right\| \leq \Delta_k$ *and* (5.15)*.*

Proof. The result follows immediately from [72, Lemma 4.2] and the fact that because $\nabla M(x^k)\mathbf{pat}_q \in \mathfrak{G}_k$, we must have $\|\nabla M(x^k)\mathbf{pat}_q\| \geq \|g^k\|$. □

Iteratively constructing the master model and identifying manifolds gives a metric $\|g^k\|$ to measure progress to Clarke stationarity, but each iteration of Algorithm 2 seeks sufficient decrease only in a manifold that lies in an approximate steepest descent direction. Note that any looping introduced by Line 18 in Algorithm 2 will terminate due to there being a finite number of manifolds in $\mathcal{B}(x^k, \Delta_k)$ (at most 3^r) and Lemma 5.3.1.

5.3.2 Generator Sets

We complete our description of our manifold sampling algorithm by showing how the set of generators $\mathfrak{G}^k$ is built so that $\mathbf{co}\left(\mathfrak{G}^k\right)$ approximates $\partial f(x^k)$. Given the definition of the Clarke subdifferential in (5.6) and the known form of the subdifferential in (5.7), we know that the extreme points of $\partial f(x)$ must be the limits of sequences of gradients at differentiable points from manifolds that are active at x. Therefore, the extreme points of $\partial f(x)$ are a subset of $\nabla F(x)\mathbf{pat}^q$ over $q \in \{1, \ldots, 3^r\}$.

An approximation

$$\nabla M(x) = \left[\nabla m^{F_1}(x), \ldots, \nabla m^{F_r}(x)\right]$$

of the Jacobian ∇F induces an approximation $\nabla M(x)\mathbf{pat}^q$ to $\nabla F(x)\mathbf{pat}^q$ for any $q \in \{1, \ldots, 3^r\}$. Since ∇F_i may not be known exactly, we relax the dependence on ∇F_i in (5.8) and consider $-\nabla m^{F_i}(x)$ and $\nabla m^{F_i}(x)$ for each i for which $F_i(x) = 0$.

This is the motivation for our first procedure for forming the generator set $\mathfrak{G}^k$. Al-

gorithm 3 initializes $\mathfrak{G}^k$ with $\nabla M(x^k)\mathbf{sign}(x^k)$ and then triples the size of $\mathfrak{G}^k$ for each i satisfying $sgn(F_i(x^k)) = 0$ and $\nabla F_i(x^k) \neq 0$. Our analysis of Algorithm 2 will show that this strategy can be used to approximate the subdifferential $\partial f(x^k)$.

Algorithm 3: Forming generator set $\mathfrak{G}^k$ using possibly active manifolds at x^k.

```
Input: x^k and ∇M(x^k)
G^k ← {∇M(x^k)sign(x^k)}
for i = 1, ..., r do
    if sgn(F_i(x^k)) = 0 then
        G^k ← G^k ∪ {G^k + ∇m^{F_i}(x^k)} ∪ {G^k − ∇m^{F_i}(x^k)}
    end
end
```

We also propose a second approach for constructing $\mathfrak{G}^k$ that uses sign patterns of points near x^k and not just $\mathbf{sign}(x^k)$. Although this approach is inspired by gradient sampling, we note that we are not approximating the gradient at any point other than x^k. We naturally extend our definition of active manifolds by saying that a manifold set $\mathcal{M}^q$ is active in a set $\mathcal{S}$ provided there exists $x \in \mathcal{S}$ such that $\mathcal{M}^q$ is active at x. We denote such a *sample set* at iteration k by $Y(x^k, \Delta_k) \subset \mathcal{B}(x^k, \Delta_k)$. This set can come, for example, from the set of points previously evaluated by the algorithm that lie within a distance Δ_k of x^k.

We can now state Algorithm 4, which constructs generators based on the set of manifolds active in $Y(x^k, \Delta_k)$. Intuitively, this additional manifold information obtained from sampling can "warn" the algorithm about sudden changes in gradient behavior that may occur within the current trust region.

Algorithm 4: Forming generator set $\mathfrak{G}^k$ using possibly active manifolds in $Y(x^k, \Delta_k)$.

```
Input: x^k, ∇M(x^k), and Y(x^k, Δ_k) = {x^k, y^2, ..., y^p}
Initialize G^k using Algorithm 3 with inputs x^k and ∇M(x^k)
for j = 2, ..., p do
    G^k = G^k ∪ ∇M(x^k)sign(y^j)
end
```

Other reasonable approaches for constructing $\mathfrak{G}^k$ exist. For our analysis, a requirement for $\mathfrak{G}^k$ is given in Assumption 5.3.2.

Assumption 5.3.2. *Given x^k and $\Delta_k > 0$, the constructed set $\mathfrak{G}^k$ satisfies*

$$\left\{ \nabla M(x^k)\mathbf{sign}(x^k) + \sum_{i:\left[\mathbf{sign}(x^k)\right]_i=0} t_i \nabla m^{F_i}(x^k) \,:\, t \in \{-1,0,1\}^r \right\} \subseteq \mathfrak{G}^k,$$

$$\mathfrak{G}^k \subseteq \left\{ \nabla M(x^k)\mathbf{sign}(y) + \sum_{i:[\mathbf{sign}(y)]_i=0} t_i \nabla m^{F_i}(x^k) \,:\, t \in \{-1,0,1\}^r, y \in \mathcal{B}(x^k;\Delta_k) \right\}.$$

Clearly, a set $\mathfrak{G}^k$ produced by Algorithm 3 or Algorithm 4 will satisfy Assumption 5.3.2. Furthermore, any generator set satisfying Assumption 5.3.2 has $|\mathfrak{G}^k| \leq 3^r$.

5.4 Analysis

We now analyze Algorithm 2.

5.4.1 Preliminaries

We first show a result linking elements in a set similar to the form of $\mathfrak{G}^k$ to the subdifferentials of f at nearby points. Subsequent results will establish cases when our construction of the generator set $\mathfrak{G}^k$ satisfies the suppositions made in the statement of the lemma.

Lemma 5.4.1. *Let Assumptions 5.1.1 and 5.3.1 hold, and let $x, y \in \mathbb{R}^n$ satisfy $\|x - y\| \leq \Delta_k$. Suppose that $T \subseteq T'$ for $T = \{\mathbf{pat}^{q_s} : s = 1, \ldots, j\}$ and $T' = \{\mathbf{pat}^{q'_{s'}} : s' = 1, \ldots, j'\}$ for*

$$\mathfrak{G} = \{\nabla M(x)p : p \in T\} \text{ and } \partial f(y) = \mathbf{co}\left\{\nabla F(y)p' : p' \in T'\right\}.$$

Then for each $g \in \mathbf{co}\,(\mathfrak{G})$, there exists $v(g) \in \partial f(y)$ satisfying

$$\|g - v(g)\| \leq (\kappa_{\mathrm{g}} + L)\Delta_k, \tag{5.17}$$

where κ_{g} and L are defined in Assumption 5.3.1 and Assumption 5.1.1, respectively.

Proof. Let $g \in \mathbf{co}\,(\mathfrak{G})$ be arbitrary. Since $\mathbf{co}\,(\mathfrak{G})$ is finitely generated (and thus compact and convex), g can be expressed as a positive convex combination of $N \leq n + 1$

of its generators due to Caratheodory's theorem. Without loss of generality (by reordering as necessary), let these generators be the first N elements in $\mathfrak{G}$. That is, there exist $\lambda_{q_1}, \ldots, \lambda_{q_N} \in (0, 1]$ with $\sum_{s=1}^{N} \lambda_{q_s} = 1$ so that

$$g = \sum_{s=1}^{N} \lambda_{q_s} \nabla M(x) \mathbf{pat}^{q_s}. \tag{5.18}$$

By supposition, $\nabla F(y_s)\mathbf{pat}^{q_s} \in \partial f(y_s)$ for $s = 1, \ldots, N$. Since $\partial f(y)$ is convex, we have that $v(g) \in \partial f(y)$, where $v(g)$ is defined as

$$v(g) = \sum_{s=1}^{N} \lambda_{q_s} \nabla F(y) \mathbf{pat}^{q_s},$$

using the same λ_{q_s} as in (5.18) for $s = 1, \ldots, N$. Observe that for each s,

$$\begin{aligned} \|\nabla M(x)\mathbf{pat}^{q_s} - \nabla F(y)\mathbf{pat}^{q_s}\| &= \|(\nabla M(x) - \nabla F(x) + \nabla F(x) - \nabla F(y))\mathbf{pat}^{q_s}\| \\ &\leq \|\nabla M(x) - \nabla F(x)\| + \|\nabla F(x) - \nabla F(y)\| \\ &\leq (\kappa_{\mathrm{g}} + L)\Delta_k. \end{aligned}$$

Applying the definitions of g and $v(g)$ and recalling that $\sum_{s=1}^{N} \lambda_{q_s} = 1$ yields the expression (5.17). □

The approximation property in Lemma 5.4.1 can be used to motivate the use of the master model gradient in (5.10); as we shall see in Section 5.4.2, descent directions for the smooth master model will eventually identify descent directions for the nonsmooth function f.

5.4.2 Analysis of Algorithm 2

The next lemma demonstrates that because the master model gradient is chosen as g^k from (5.10), the sufficient decrease condition in (5.15) ensures a successful iteration, provided Δ_k is sufficiently small.

Lemma 5.4.2. *Let Assumptions 5.1.1 and 5.3.1 hold. If*

$$\Delta_k < \min\left\{\frac{\kappa_{\mathrm{d}}(1-\eta_1)}{4\kappa_{\mathrm{f}}}, \eta_2\right\}\|g^k\|, \tag{5.19}$$

then iteration k of Algorithm *2 is successful.*

Proof. Notice that, whenever $g^k \neq 0$, the bound on Δ_k is positive by the algorithmic parameter assumptions in Algorithm 2 and that $\Delta_k < \eta_2\|g^k\|$ ensures that the iteration is acceptable. Suppressing superscripts on x^k, s^k, and w^k for space, the definition of ρ_k in (5.14) yields

$$\begin{aligned}
&|\rho_k - 1| \\
&= \left|\frac{\langle F(x), \mathbf{sign}(x+s)\rangle - \langle F(x+s), \mathbf{sign}(x+s)\rangle}{\langle M(x), \mathbf{sign}(x+s)\rangle - \langle M(x+s), \mathbf{sign}(x+s)\rangle} - 1\right| \\
&= \left|\frac{\langle F(x) - M(x), \mathbf{sign}(x+s)\rangle - \langle F(x+s) - M(x+s), \mathbf{sign}(x+s)\rangle}{\langle M(x) - M(x+s), \mathbf{sign}(x+s)\rangle}\right|.
\end{aligned} \tag{5.20}$$

The numerator of (5.20) satisfies

$$\begin{aligned}
&|\langle F(x) - M(x), \mathbf{sign}(x+s)\rangle - \langle F(x+s) - M(x+s), \mathbf{sign}(x+s)\rangle| \\
&\leq |\langle F(x) - M(x), \mathbf{sign}(x+s)\rangle| + |\langle F(x+s) - M(x+s), \mathbf{sign}(x+s)\rangle| \\
&\leq 2\kappa_{\mathrm{f}}\Delta_k^2.
\end{aligned} \tag{5.21}$$

By construction, the denominator of (5.20) always satisfies (5.15). Therefore, using (5.21) and (5.15) in (5.20) yields

$$|\rho_k - 1| \leq \frac{4\kappa_{\mathrm{f}}\Delta_k^2}{\kappa_{\mathrm{d}}\|g^k\| \min\left\{\Delta_k, \frac{\|g^k\|}{\kappa_{\mathrm{mh}}}\right\}} \leq \frac{4\kappa_{\mathrm{f}}\Delta_k}{\kappa_{\mathrm{d}}\|g^k\|} < 1 - \eta_1,$$

where the second inequality is implied by the algorithmic parameter assumption $\frac{1}{\eta_2} > \kappa_{\mathrm{mh}}$ and the last inequality is a result of (5.19). Thus, $\rho_k > \eta_1$, and iteration k is successful. □

The next result shows that the trust-region radius converges to zero.

Lemma 5.4.3. *Let Assumptions 5.1.1 and 5.3.1 hold. If $\{x^k, \Delta_k\}$ is generated by* Algorithm *2, then $\Delta_k \to 0$.*

Proof. On successful iterations k, $\rho_k > \eta_1$ and thus,

$$\begin{aligned}
\langle F(x) - F(x+s), \mathbf{sign}(x+s)\rangle &> \eta_1 \left(\langle M(x) - M(x+s), \mathbf{sign}(x+s)\rangle\right) \\
&\geq \eta_1 \frac{\kappa_{\mathrm{d}}}{2} \|g^k\| \min\left\{\Delta_k, \frac{\|g^k\|}{\kappa_{\mathrm{mh}}}\right\} \\
&\geq \eta_1 \frac{\kappa_{\mathrm{d}}}{2} \|g^k\| \Delta_k \\
&> \frac{\eta_1 \kappa_{\mathrm{d}}}{2\eta_2} \Delta_k^2,
\end{aligned}$$

where the second inequality follows from (5.15), and the last two inequalities follow from the algorithmic parameter assumption $\frac{1}{\eta_2} > \kappa_{\mathrm{mh}}$ and acceptability of all successful iterations. If there are infinitely many successful iterations, let $\{k_j\}$ index them. Notice that on any iteration,

$$\langle F(x^k), \mathbf{sign}(x^k + s^k)\rangle \leq \langle F(x^k), \mathbf{sign}(x^k)\rangle = f(x^k),$$

since, for any i, $F_i(x^k)[\mathbf{sign}(x^k)]_i = |F_i(x^k)| \geq t_i F_i(x^k)$ for all $t_i \in \{-1, 0, 1\}$.

Since f is bounded below by zero by Assumption 5.1.1 and $f(x^k)$ is nonincreasing in k, having infinitely many successful iterations implies that

$$\begin{aligned}
\infty > \sum_{j=0}^{\infty} f(x^{k_j}) - f(x^{k_j} + s^{k_j}) &> \sum_{j=0}^{\infty} \langle F(x^{k_j}) - F(x^{k_j} + s^{k_j}), \mathbf{sign}(x^{k_j} + s^{k_j})\rangle \\
&> \sum_{j=0}^{\infty} \frac{\eta_1 \kappa_{\mathrm{d}}}{2} \left\|g^{k_j}\right\| \Delta_{k_j} \\
&> \sum_{j=0}^{\infty} \frac{\eta_1 \kappa_{\mathrm{d}}}{2\eta_2} \Delta_{k_j}^2. \qquad (5.22)
\end{aligned}$$

Thus, $\Delta_{k_j} \to 0$ provided $\{k_j\}$ is an infinite subsequence of successful iterations. Since Δ_k increases by γ_{inc} on successful iterations, for any successful iterate k_i, $\gamma_{\mathrm{inc}} \Delta_{k_i} \geq \Delta_j \geq \Delta_{k_{i+1}}$ for all $k_i < j \leq k_{i+1}$. Therefore, $\Delta_k \to 0$ if the number of successful iterations is infinite.

If there are only finitely many successful iterations, then there is a last successful

iteration $k_{j'}$, and the update rules of the algorithm will monotonically decrease Δ_k on all iterations $k > k_{j'}$.

Regardless of whether Algorithm 2 has an infinite or a finite number of successful iterations, $\Delta_k \to 0$. □

The next lemma is a liminf-type result for the master model gradients.

Lemma 5.4.4. *Let Assumptions 5.1.1 and 5.3.1 hold. If $\{x^k, \Delta_k\}$ is generated by* Algorithm *2, then for all $\epsilon > 0$, there exists a $k(\epsilon)$ such that $\|g^{k(\epsilon)}\| \leq \epsilon$. That is,*

$$\liminf_{k\to\infty} \|g^k\| = 0.$$

Proof. To arrive at a contradiction, suppose that there exist j and $\epsilon > 0$ so that for all $k \geq j$, $\|g^k\| \geq \epsilon$. By Lemma 5.4.2, since Algorithm 2 requires that $\nabla M(x^k)\mathbf{sign}(x^k + s^k) \in \mathfrak{G}^k$ before s^k can possibly be accepted, any iteration satisfying $\Delta_k \leq C\|g^k\|$ will be successful, where

$$C = \min\left\{\frac{\kappa_{\mathrm{d}}(1-\eta_1)}{4\kappa_{\mathrm{f}}}, \eta_2\right\}.$$

Hence, by the contradiction hypothesis, any $k \geq j$ satisfying $\Delta_k \leq C\epsilon$ is guaranteed to be successful and $\Delta_{k+1} = \gamma_{\mathrm{inc}}\Delta_k \geq \Delta_k$. Therefore $\Delta_k \geq \gamma_{\mathrm{dec}}C\epsilon$ for all k, contradicting Lemma 5.4.3. □

Before showing that every cluster point of $\{x^k\}$ is a Clarke stationary point, we recall basic terms and a theorem.

Motivated by the subdifferential operator $\partial f(x)$, we first formalize the notion of a limit superior of a set mapping (i.e., one that maps a vector to a set of vectors). For a set mapping $D : \mathbb{R}^n \to \mathbb{R}^n$, the *limit superior* of D as $x \to \bar{x}$ is defined by the set mapping

$$\limsup_{x\to\bar{x}} D(x) = \{y : \exists\{x^k : k \geq 1\} \to \bar{x} \text{ and } \{y^k : k \geq 1\} \to y \text{ with } y^k \in D(x^k)\}.$$

A set mapping D is said to be *outer semicontinuous at* $\bar{x}$ provided

$$\limsup_{x\to\bar{x}} D(x) = D(\bar{x}).$$

The following result is given as Proposition 7.1.4 in [28].

Theorem 5.4.1. *If a function $f : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuous, then the set mapping $\partial f(x)$ is everywhere outer semicontinuous.*

Lemma 5.4.5. *Let Assumptions 5.1.1–5.3.2 hold and take $\{x^k, \Delta_k, g^k\}$ to be a sequence generated by* Algorithm *2. For any subsequence of acceptable iterations $\{k_j\}$ such that*

$$\lim_{j \to \infty} \|g^{k_j}\| = 0,$$

and $\left\{x^{k_j}\right\} \to x^*$ for some point x^*, then $0 \in \partial f(x^*)$.

Proof. Since $\Delta_k \to 0$ by Lemma 5.4.3 and $\left\{x^{k_j}\right\}$ converges to x^*, for k sufficiently large, only manifolds active at x^* are in $\mathfrak{G}^k$ by Assumption 5.3.2. Setting T (in Lemma 5.4.1) to be the sign patterns in $\mathfrak{G}^k$ and T' to be the sign patterns necessary for building $\partial f(x^*)$, Assumption 5.3.2, Lemma 5.4.3, and Theorem 2.3.9 from [18] thus guarantee $T \subseteq T'$ when k is sufficiently large. Therefore, by Lemma 5.4.1, there exists $v(g^{k_j}) \in \partial f(x^*)$ for each g^{k_j} so that

$$\|g^{k_j} - v(g^{k_j})\| \leq (\kappa_{\mathrm{g}} + L)\Delta_{k_j}.$$

Thus, by the acceptability of every iteration indexed by k_j,

$$\|g^{k_j} - v(g^{k_j})\| \leq (\kappa_{\mathrm{g}} + L)\eta_2 \|g^{k_j}\|,$$

and so

$$\|v(g^{k_j})\| \leq (1 + (\kappa_{\mathrm{g}} + L)\eta_2)\, \|g^{k_j}\|.$$

Since $\left\|g^{k_j}\right\| \to 0$ by assumption, therefore $v(g^{k_j}) \to 0$. Since ∂f is everywhere outer semicontinuous by Theorem 5.4.1, $0 \in \partial f(x^*)$. □

We now prove the promised result.

Theorem 5.4.2. *Let Assumptions 5.1.1–5.3.2 hold. If x^* is a cluster point of a sequence $\{x^k\}$ generated by* Algorithm *2, then $0 \in \partial f(x^*)$.*

Proof. Suppose that there are only finitely many successful iterations, with k' being the last.

To establish a contradiction, suppose $0 \notin \partial f(x^{k'})$. By the continuity of each component F_i granted by Assumption 5.1.1, there exists $\bar{\Delta} > 0$ so that for all $\Delta \in [0, \bar{\Delta})$, the manifold sets active in $\mathcal{B}(x^{k'}, \bar{\Delta})$ are precisely the manifold sets active at $x^{k'}$; that is,

$$\left\{ \lim_{y^j \to x^{k'}} \mathbf{sign}(y^j) : \lim_{j \to \infty} y^j = x^{k'},\ \{y^j : j \geq 1\} \subset \mathcal{M}^q \right\} = \left\{ \mathbf{sign}(y) : y \in \mathcal{B}(x^{k'}, \Delta) \right\}.$$

for all $\Delta \leq \bar{\Delta}$.

Since every iteration after k' is assumed to be unsuccessful, $\Delta_{k'}$ decreases by a factor of γ_{dec} in each subsequent iteration and there exists a least $k'' \geq k'$ so that $\Delta_{k''} < \bar{\Delta}$. Therefore, by Assumption 5.3.2, $\nabla M(x^k)\mathbf{sign}(x^k + s^k) \in \mathfrak{G}^k$ holds the first time Line 13 of Algorithm 2 is reached in iteration $k \geq k''$. Consequently, the conditions for Lemma 5.4.1 hold; and thus, for each $k \geq k''$, there exists $v(g^k) \in \partial f(x^{k'})$ so that $\|v(g^k) - g^k\| \leq (\kappa_{\text{g}} + L)\Delta_k$. By supposition, since $0 \notin \partial f(x^{k'})$, there is a nonzero minimum-norm element $v^* \in \partial f(x^{k'})$. We thus conclude the following:

$$\|g^k\| \geq \|v(g^k)\| - (\kappa_{\text{g}} + L)\Delta_k \geq \|v^*\| - (\kappa_{\text{g}} + L)\Delta_k \qquad \text{for all } k \geq k''. \tag{5.23}$$

Since every iteration after k'' is unsuccessful, Δ_k will decrease by a factor of γ_{dec} and $x^{k+1} = x^{k'}$ for each $k \geq k''$.

Define the constant

$$c = \|v^*\| \min \left\{ \frac{(1 - \eta_1)}{4\frac{\kappa_{\text{f}}}{\kappa_{\text{d}}} + (\kappa_{\text{g}} + L)(1 - \eta_1)}, \frac{1}{\kappa_{\text{mh}} + \kappa_{\text{g}} + L} \right\}.$$

By Lemma 5.4.2, success is guaranteed within $t = \left\lceil \log_{\gamma_{\text{dec}}} \frac{c}{\Delta_{k''}} \right\rceil$ many iterations after iteration k'' since (5.23) and the definition of c imply that

$$\Delta_{k''+t} \leq \min \left\{ \frac{\kappa_{\text{d}}(1 - \eta_1)}{4\kappa_{\text{f}}}, \eta_2 \right\} \|g^{k''+t}\|.$$

This is a contradiction, thus proving the result when there are finitely many successful iterations.

Now suppose there are infinitely many successful iterations. We will demonstrate that there exists a subsequence of successful iterations $\{k_j\}$ that simultaneously satisfies both

$$x^{k_j} \to x^* \text{ and } \|g^{k_j}\| \to 0.$$

Suppose first that $x^k \to x^*$. Then, every subsequence $x^{k_j} \to x^*$, and so we can use the sequence of subgradient approximations $\{g^{k_j}\}$ from Lemma 5.4.4, and we have the desired subsequence.

Now, suppose $x^k \not\to x^*$. We will show that $\liminf \max\{\|x^k - x^*\|, \|g^k\|\} = 0$. We proceed by contradiction. The contradiction hypothesis, along with the definition of x^* being a cluster point of $\{x^k\}$, implies that there exists $\bar{\nu} > 0$ and iteration $\bar{k}$, so that for the infinite set $K = \{k : k \geq \bar{k}, \|x^k - x^*\| \leq \bar{\nu}\}$, $\left\{x^k\right\}_{k \in K} \to x^*$ and $\|g^k\| > \bar{\nu}$ for all $k \in K$. From (5.22), we have that

$$\frac{\eta_1 \kappa_{\mathrm{d}}}{2} \sum_{k \in K} \|g^k\| \|x^{k+1} - x^k\| \leq \frac{\eta_1 \kappa_{\mathrm{d}}}{2} \sum_{k=0}^{\infty} \|g^k\| \|x^{k+1} - x^k\| < \infty, \tag{5.24}$$

since on successful iterations, $\|x^{k+1} - x^k\| \leq \Delta_k$, while on unsuccessful iterations, $\|x^{k+1} - x^k\| = 0$. Since $\|g^k\| > \bar{\nu}$ for all $k \in K$, then we conclude from (5.24) that

$$\sum_{k \in K} \|x^{k+1} - x^k\| < \infty. \tag{5.25}$$

Since $x^k \not\to x^*$, there exists some $\hat{\nu} > 0$ so that for any $k_{\text{start}} \in K$ satisfying $\|x^{k_{\text{start}}} - x^*\| \leq \bar{\nu}$, there a first index $k_{\text{end}} > k_{\text{start}}$ satisfying $\|x^{k_{\text{end}}} - x^{k_{\text{start}}}\| > \hat{\nu}$ and $\{k_{\text{start}}, k_{\text{start}} + 1, \ldots, k_{\text{end}} - 1\} \subset K$.

By (5.25), for $\hat{\nu}$ there exists $N \in \mathbb{N}$ such that

$$\sum_{\substack{k \in K \\ k \geq N}} \left\| x^{k+1} - x^k \right\| \leq \hat{\nu}.$$

Taking $k_{\text{start}} \geq N$, by the triangle inequality, we have

$$\hat{\nu} < \|x^{k_{\text{end}}} - x^{k_{\text{start}}}\| \leq \sum_{i \in \{k_{\text{start}}, k_{\text{start}}+1, \ldots, k_{\text{end}}-1\}} \|x^{i+1} - x^i\| \leq \sum_{\substack{k \in K \\ k \geq N}} \left\| x^{k+1} - x^k \right\| \leq \hat{\nu}.$$

Thus, $\hat{\nu} < \hat{\nu}$, a contradiction, and therefore $\liminf \max\{\|x^k - x^*\|, \|g^k\|\} = 0$.

Therefore, by Lemma 5.4.5, in either the case where $x^k \to x^*$ or $x^k \not\to x^*$, there exists a subsequence of subgradients satisfying $\|v(g^{k_j})\| \to 0$ and $x^{k_j} \to x^*$. Since $\partial f(x^*)$ is outer semicontinuous, we have that $0 \in \partial f(x^*)$. □

5.4.3 Concerning Termination Certificates

One would hope that (on acceptable iterations), a small master model gradient norm would signal that necessary conditions for proximity to stationarity are satisfied, analogous to how a small gradient norm serves as such a signal in smooth optimization. Although this is not always so, we provide exact theoretical conditions under which a similar statement holds for Algorithm 2. This approach emulates a stopping criterion used in [39], which likewise cannot generally be shown to be a necessary condition of proximity to stationarity.

Lemma 5.4.6. *Let Assumptions 5.1.1 and 5.3.1 hold. Suppose that at iteration k, Line 25 of* Algorithm *2 is reached with $\|g^k\| < \epsilon$ for some $\epsilon > 0$ and also let $\mathfrak{G}^k = \{\nabla M(x^k)\mathbf{pat}^{q_s} : s = 1, \ldots, j\}$ be the generator set at iteration k. Additionally, suppose there exists $y \in \mathcal{B}(x^k, \Delta_k)$ so that $\partial f(y) = \mathbf{co}\left\{\nabla F(y)\mathbf{pat}^{q'_{s'}} : s' = 1, \ldots, j'\right\}$ with $\{\mathbf{pat}^{q_1}, \ldots, \mathbf{pat}^{q_j}\} \subseteq \{\mathbf{pat}^{q'_1}, \ldots, \mathbf{pat}^{q'_{j'}}\}$. Then,*

$$\min_{v \in \partial f(y)} \|v\| < (1 + (\kappa_{\text{g}} + L)\eta_2)\epsilon,$$

where κ_{g} is as in Assumption 5.3.1 and L is as in Assumption 5.1.1.

Proof. By the algorithmic parameter assumptions in Algorithm 2, $\eta_2 > 0$. Since Line 25 is reached with $\|g^k\| < \epsilon$, it must be that $\frac{\Delta_k}{\eta_2} \leq \|g^k\| < \epsilon$. By Lemma 5.4.1 and the

suppositions, there exists $v(g^k) \in \partial f(y)$ so that

$$\|g^k - v(g^k)\| \leq (\kappa_g + L)\Delta_k < (\kappa_g + L)\eta_2\epsilon.$$

Applying the triangle inequality yields

$$\|v(g^k)\| < (1 + (\kappa_g + L)\eta_2)\epsilon,$$

from which the desired result follows. □

We note that the assumptions that the iteration is acceptable and that y lies within Δ_k of x^k directly tie the result to both the master model gradient $\|g^k\|$ and the trust-region radius Δ_k. A termination certificate consisting of these two quantities is analogous to the "optimality certificates" used in [11]. In both cases, the certificate can be interpreted as indicating that two of three necessary conditions are satisfied so that there is a point $y \in \mathcal{B}(x^k, \Delta_k)$ so that an element of $\partial f(y)$ is as small in norm as suggested in Lemma 5.4.6. The third necessary condition, which is not as straightforward to check, is that the algorithm's iterates have become sufficiently clustered around y so that the manifolds active at y are a superset of the manifolds active in $\mathcal{B}(x^k, \Delta_k)$.

We remark here that this dependence on knowing all the manifolds active in a given trust region is what makes a straightforward analysis of rates based on $\|g_k^f\|$ elusive. Therefore, a comparison of the theoretical worst-case complexity of Algorithm 2 with the rates proven for the nonsmooth methods in [32] is currently elusive.

5.5 Manifold Sampling as a Stochastic Algorithm

Thus far, no restrictions have been placed on the sample set $Y(x^k, \Delta_k)$, apart from its containment in $\mathcal{B}(x^k, \Delta_k)$. In this section, we consider what happens when $Y(x^k, \Delta_k)$ includes stochastically sampled points, a strategy that results in Section 5.6 show is fruitful when $\mathfrak{G}^k$ is built by using Algorithm 4.

If random points are added to $Y(x^k, \Delta_k)$, the sequence of generator sets $\{\mathfrak{G}^k : k \geq 1\}$

will be a realization of a random variable denoted $\{\tilde{\mathfrak{G}}^k : k \geq 1\}$. Consequently, the algorithm will be inherently stochastic; the sequence of iterates produced by Algorithm 2 will be random variables $\{\tilde{x}^k : k \geq 1\}$ with realizations $\{x^k : k \geq 1\}$. Similarly, we denote by $\{\tilde{\Delta}_k : k \geq 1\}$, $\{\tilde{s}^k : k \geq 1\}$, and $\{\tilde{g}^k : k \geq 1\}$ sequences of random trust-region radii, trial steps, and master model gradients with respective realizations $\{\Delta_k : k \geq 1\}$, $\{s^k : k \geq 1\}$, and $\{g^k : k \geq 1\}$.

We show in Theorem 5.5.1 that the results from Section 5.4 hold for *any* realization of Algorithm 2. Note that Assumption 5.1.1 is unaffected by stochasticity in $Y(x^k, \Delta_k)$ (and $\mathfrak{G}^k$ is similarly unaffected on any iteration). In particular, we note that Assumption 5.3.1 ensures that the quality of the component models holds in a deterministic fashion. Assumption 5.3.2 has the stochastic analogue of assuming that every iterate in any realization satisfies the deterministic Assumption 5.3.2. This is nonrestrictive since Assumption 5.3.2 requires that $\mathfrak{G}^k$ satisfy conditions depending not on $Y(x^k, \Delta_k)$ but only on the sign patterns in $\mathcal{B}(x^k, \Delta_k)$, which is a deterministic set at any iteration.

Theorem 5.5.1. *Let X^* denote the union of all cluster points x^* over all realizations $\{(x^k, \Delta_k, s^k, \mathfrak{G}^k, g^k) : k \geq 1\}$ in the σ-algebra generated by $\{(\tilde{x}^k, \tilde{\Delta}_k, \tilde{s}^k, \tilde{\mathfrak{G}}^k, \tilde{g}^k) : k \geq 1\}$. Let Assumptions 5.1.1 and 5.3.1 hold, let Assumption 5.3.2 hold for each (x^k, Δ_k) in any realization $\{(x^k, \Delta_k) : k \geq 1\}$, and let* Algorithm *2 be initialized with $(\tilde{x}^0, \tilde{\Delta}_0) = (x^0, \Delta_0)$. Then, for every $x^* \in X^*$, $0 \in \partial f(x^*)$.*

Proof. The proof follows the same argument as in Section 5.4.

Let $\{(x^k, \Delta_k, s^k, \mathfrak{G}^k, g^k) : k \geq 1\}$ be an arbitrary realization of the random sequence $\{(\tilde{x}^k, \tilde{\Delta}_k, \tilde{s}^k, \tilde{\mathfrak{G}}^k, \tilde{g}^k) : k \geq 1\}$ produced by the stochastic algorithm.

Lemma 5.4.1 is independent of the realization, and Theorem 5.4.1 holds independently of Algorithm 2. Lemma 5.4.2 holds deterministically for any k where Δ_k and $\|g^k\|$ (produced by $\mathfrak{G}^k$) satisfy $\Delta_k < C\|g^k\|$.

Lemma 5.4.3 depends only on f being bounded below (a result of Assumption 5.1.1), and thus we get $\Delta_k \to 0$ for the arbitrary realization.

Lemma 5.4.4 holds if $Y(x^k, \Delta_k)$ is stochastic, thereby producing stochastic $\mathfrak{G}^k$, because the realization $\{(\Delta_k, g^k) : k \geq 1\}$ having $\liminf_{k\to\infty} \|g^k\| \neq 0$ would similarly contradict

$\Delta_k \to 0$ (Lemma 5.4.3).

We can now prove the theorem. Suppose $\{x^k : k \geq 1\}$ has a cluster point x^*. Then, having proved that all the lemmata hold for the arbitrary realization, a direct application of Theorem 5.4.2 to that particular realization gives us that $0 \in \partial f(x^*)$. Since the realization of $\{(\tilde{x}^k, \tilde{\Delta}_k, \tilde{s}^k, \tilde{\mathfrak{G}}_k, \tilde{g}^k) : k \geq 1\}$ was arbitrary, we have shown the desired result. □

5.6 Numerical Results

We now examine the performance of variations of the manifold sampling algorithm outlined in Algorithm 2. Throughout this section, we use $x^{(j,p,s)}$ to denote the jth point evaluated on a problem p by an optimization solver s. For derivative-based versions of Algorithm 2, such points correspond solely to the trust-region subproblem solutions (Line 11) and points possibly sampled when constructing the generator set (Line 5); for derivative-free versions, evaluated points may additionally include evaluations performed to ensure that the component models are fully linear in the current trust region (Line 4). We drop the final superscript (s) when the point is the same for all solvers.

5.6.1 Implementations

In our first tests, we focus on the derivative-free case, when only zeroth-order information (function values) of F is provided to a solver; we view such problems as a more challenging test of the manifold sampling algorithm, since the component model approximations are not directly obtained from derivative information.

In our implementations of Algorithm 2, the sampling set $Y(x^k, \Delta_k)$ is used for construction of both the component models and, when using Algorithm 4, the generator sets. All our implementations employ linear models, m^{F_i}, of each component function F_i. The linear models are constructed so that m^{F_i} interpolates F_i at x^k and is the least-squares regression model for the remainder of the sampling set, $Y(x^k, \Delta_k) \backslash x^k$. At the beginning of iteration k, we set $Y(x^k, \Delta_k)$ to be all points previously evaluated by the algorithm that lie in $\mathcal{B}(x^k, \Delta_k)$. If this results in an underdetermined interpolation (i.e., $\text{rank}\big(Y(x^k, \Delta_k) - x^k\big) < n$), then additional points are added to $Y(x^k, \Delta_k)$ as described below.

We tested four variants of Algorithm 2, which differ from one another in how they construct the generator set $\mathfrak{G}^k$ and how they add points to the sampling set $Y(x^k, \Delta_k)$:

Center Manifold Sampling (CMS): Uses Algorithm 3 to build the generator set $\mathfrak{G}^k$; this generator set does not depend on the sampling set $Y(x^k, \Delta_k)$, which is used solely for constructing the component models. For building these models, the set of scaled coordinate directions, $\{x^k + \Delta_k e_1, \ldots, x^k + \Delta_k e_n\}$, are added to the sample set $Y(x^k, \Delta_k)$ in cases of underdetermined interpolation.

Greedy Deterministic Manifold Sampling (GDMS): Uses Algorithm 4 to build the generator set $\mathfrak{G}^k$. Additional points are not added to the sample set $Y(x^k, \Delta_k)$ unless the linear regression is underdetermined. In the underdetermined case, $n - \text{rank}(Y(x^k, \Delta_k) - x^k)$ directions D in the null space of $Y(x^k, \Delta_k) - x^k$ are generated by means of a (deterministic) QR factorization. After evaluating F along these scaled directions, the associated points $x^k + \Delta_k D$ are added to the sample set $Y(x^k, \Delta_k)$.

Deterministic Manifold Sampling (DMS): Uses Algorithm 4 to build the generator set and adds scaled coordinate directions, $\{x^k + \Delta_k e_1, \ldots, x^k + \Delta_k e_n\}$, to the sample set $Y(x^k, \Delta_k)$ every iteration.

Stochastic Manifold Sampling (SMS): Uses Algorithm 4 to build the generator set and adds a set of n points randomly generated from a uniform distribution on $\mathcal{B}(x^k, \Delta_k)$ to the sample set $Y(x^k, \Delta_k)$ every iteration.

The strategy used to add points to the sample set $Y(x^k, \Delta_k)$ in the deterministic variants ensures the full linearity of the models required in Assumption 5.3.1. For the stochastic variant SMS, however, the realized sample set $Y(x^k, \Delta_k)$ results in models that do not necessarily satisfy Assumption 5.3.1. Consequently, Theorem 5.5.1 may not hold for our implementation since such a sample set does not guarantee that the realized models are fully linear (see, e.g., [21, 48]).

In all cases, the weights defining the master model in Line 7 of Algorithm 2 are calculated by solving the quadratic program (5.11) via the subproblem solver used in [22], which is based on a specialized active set method proposed in [44].

We compared the above variants with a modified version of the minimax method of Grapiglia et al. [36], which we denote GYY. We adjusted the code used in [36] to solve ℓ_1-problems by changing the nonsmooth subproblem linear program (5.9). We also adjusted stopping tolerances to prevent early termination: we decreased the minimum trust-region radius to 10^{-32} and removed the default criterion of stopping after 10 successive iterations without a decrease in the objective.

As a baseline, we also tested two trust-region algorithms for smooth optimization. The codes L-DFOTR and Q-DFOTR are implementations of the algorithm described in [21] using, respectively, linear and quadratic regression models defined by an appropriately sized, deterministic sample set. These implementations may not converge to a stationary point since they assume a smooth objective function, but they serve as important comparators since they are more efficient at managing their respective sample sets.

By design, all seven of the codes tested employ a trust-region framework, and thus the parameters across the methods can be set equal. The parameter constants were selected to be $\Delta_0 = \max\left\{1, \left\|x^{(0,p)}\right\|_\infty\right\}$, $\eta_1 = 0.25$, $\eta_2 = 1$, $\gamma_{\text{dec}} = 0.5$, and $\gamma_{\text{inc}} = 2$.

5.6.2 Test Problems

We consider the ℓ_1 test problems referred to as the "piecewise smooth" test set in [55]. This synthetic test set was selected in part because of the availability of the Jacobian $\nabla F(x)$ for each problem, and thus the subdifferential in (5.7); this is useful for benchmarking purposes. The set is composed of 53 problems of the form (5.1) ranging in dimension from $n = 2$ to $n = 12$, with the number of component functions, r, ranging from n to 65.

A standard starting point, $x^{(0,p)}$, is provided for each problem p in the test set. We note that the objective f is nondifferentiable at $x^{(0,p)}$ for five problems (numbers 9, 10, 29, 30, and 52).

For all test problems, there is neither a guarantee that there is a unique minimizer x^* with $0 \in \partial f(x^*)$ nor a guarantee that $f(x^*) = f(y^*)$ for all x^*, y^* with $0 \in \partial f(x^*) \cap \partial f(y^*)$.

A budget of $1000(n + 1)$ function evaluations was given to each solver for each n-dimensional problem. Solvers were terminated short of this budget only when Δ_k fell

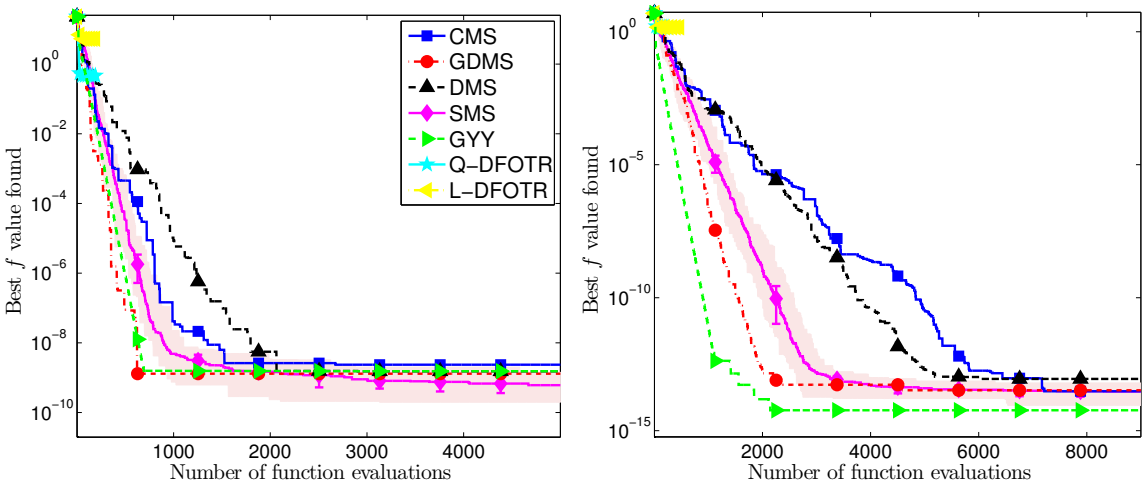

Figure 5.1: Sample trajectories of function values on problem 11 *(left)*, which has $n = r = 4$, and problem 52 *(right)*, which has $n = r = 8$. In SMS, 30 runs were performed: the upper band shows the largest function value obtained at the indicated number of function evaluations and the lower band shows the least function value obtained; the median values are indicated in the line segment connecting the 25th and 75th quantiles of the function values.

below 10^{-32}. We note, however, that for virtually every solver and problem, no successful iterations were found after Δ_k fell below 10^{-18}; this result is unsurprising given that the experiments were run in double precision.

Figure 5.1 shows typical trajectories of the best function value found on two problems where $\min f(x) = 0$. The sole stochastic solver (SMS) was run 30 times; Figure 5.1 shows that there is little variability in the value of the solution found by SMS across the 30 instances (on these two problems). This is not the case in Figure 5.2 (left), which shows the trajectory on problem 31; here we see that at least one instance of SMS finds a best function value different from that of the majority of SMS instances. In each of these instances, the smooth solvers L-DFOTR and Q-DFOTR struggle to find solutions with function values comparable to those found by the nonsmooth solvers.

5.6.3 Measuring Stationarity

The behavior seen in Figure 5.2 (left) suggests that the function values found by a solver may not indicate whether the solver has found a stationary point. We now measure the ability of a solver to identify points close to Clarke stationarity.

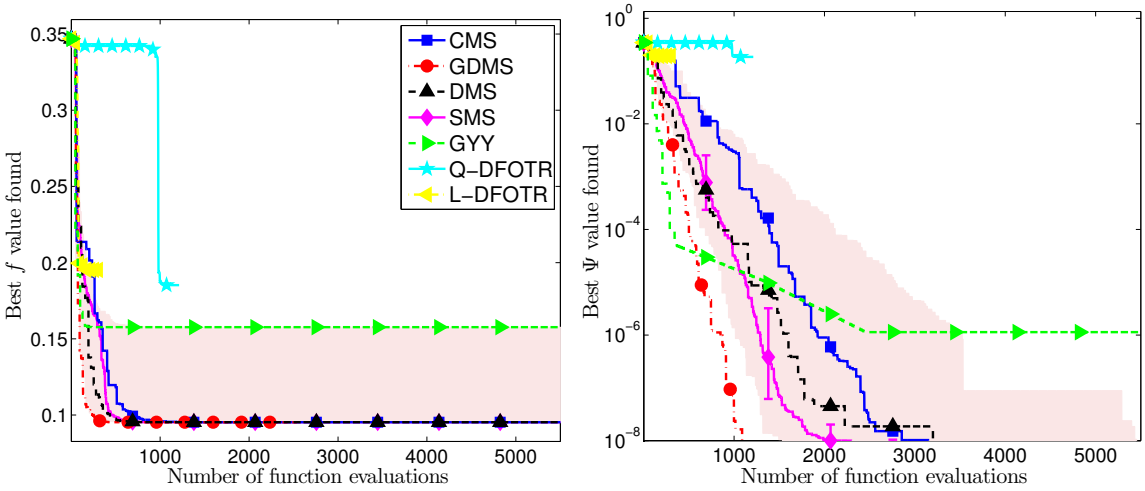

Figure 5.2: Best function value *(left)* and stationary measure $\Psi(x)$ *(right)* found in terms of the number of function evaluations performed for problem 31, a problem with $n = 8$ dimensions and $r = 8$ components. The stationary measure indicates that all instances of SMS find a stationary point, despite the function values associated with these stationary points being different.

Lemma 5.4.6 does not guarantee that $(\Delta_k, \|g^k\|)$ provides a measure of stationarity. Instead, we will employ the stationarity measure used for nonsmooth composite optimization in [74] and more recently in [36]. This measure considers the maximum decrease obtained from directional linearizations of f at x,

$$\Psi(x) = \max_{d:\|d\|\leq 1} f(x) - \|F(x) + \nabla F(x)^\top d\|_1. \tag{5.26}$$

From the fact that f is Clarke regular (see, e.g., [18]), a cluster point x^* of $\{x^k\}$ being Clarke stationary is equivalent to the condition $\liminf_{k\to\infty} \Psi(x^k) = 0$. Such a stationary measure is also readily computed for our benchmark problems since the Jacobian ∇F is known. For example, when the Euclidean norm on d in (5.26) is changed to an ℓ_∞ norm, $\Psi(x^k)$ can be obtained by solving the linear optimization problem

$$\text{minimize}_{d,s} \left\{ e^\top s : \; s \geq F(x^k) + \nabla F(x^k)^\top d, \; s \geq -F(x^k) - \nabla F(x^k)^\top d, \; d \in [-1, 1]^n \right\}.$$

The importance of using the stationary measure Ψ is highlighted in Figure 5.2, where we see the performance of seven algorithms on problem 31. Most of the manifold sampling

implementations find the same (and largest) amount of function-value decrease, but GYY and an SMS instance converge to a point with a relatively worse function value. Even so, these points all have similar stationary behavior. L-DFOTR and Q-DFOTR fail to find a stationary point.

5.6.4 Measuring Performance across the Set

For comparing the performance of algorithms across the entire test set, we use the data profiles described in [55]. Let S denote the set of solvers we wish to compare, and let P denote the set of test problems. Let $t_{p,s}$ denote the number of function evaluations required for solver $s \in S$ to satisfy a convergence criterion on problem $p \in P$. We use the convention that $t_{p,s} = \infty$ if the convergence criterion is not satisfied within the budget of evaluations.

For $\kappa \geq 0$, the data profile for solver s is then defined by

$$d_s(\kappa) = \frac{1}{|P|} \left| \{p \in P : t_{p.s} \leq \kappa(n_p + 1)\} \right|,$$

where n_p is the dimension of problem p.

We first examine the convergence criterion in [55], which is based on the best function value found by an algorithm. In particular, given a tolerance $\tau > 0$, we will say that solver s has converged on problem p when an $x^{(j,p,s)} \in \mathbb{R}^{n_p}$ has been found such that

$$f(x^{(j,p,s)}) \leq f_p + \tau(f(x^{(0,p)}) - f_p), \tag{5.27}$$

where f_p is the least function value obtained across all evaluations of all solvers in S for problem p and where $x^{(0,p)}$ is an initial point common to all solvers. The parameter τ determines how accurate one expects a solution to be in terms of the achievable decrease $f(x^{(0,p)}) - f_p$.

Figure 5.3 (left) shows that all manifold sampling implementations and GYY successfully find points with function values better than 99.9% of the best-found decrease for over 90% of the problems. For the smaller τ, the solvers' performances are more distinguishable. SMS finds decrease at least as good as $(1 - 10^{-7})\%$ of the best-performing method on 75%

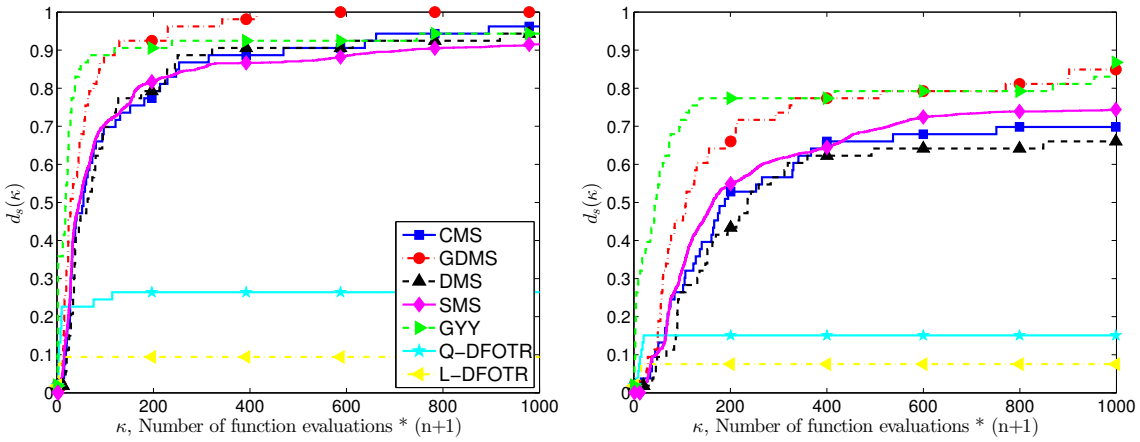

Figure 5.3: Data profiles based on the function value convergence measure (5.27) for $\tau = 10^{-3}$ (left) and $\tau = 10^{-7}$ (right).

of the problems, while GDMS and GYY do so for 85% of the problems. The relative success of GDMS over the other deterministic solvers highlights the importance of judiciously using the budget of function evaluations. The quick plateau behavior for the smooth solvers L-DFOTR and Q-DFOTR indicate that these solvers are efficient on the problems that they are able to solve.

Our next convergence criterion relates to the stationarity measure Ψ defined in (5.26). Given a tolerance $\tau > 0$, we say that convergence has occurred when

$$\Psi(x^{(j,p,s)}) \leq \tau \Psi(x^{(0,p)}). \tag{5.28}$$

Using (5.28) to test for convergence in Figure 5.4, we gain additional insight into the performance of the solvers. Figure 5.4 (left) shows that Q-DFOTR and L-DFOTR do not find points with Ψ values less than one-thousandth of the stationary measure at $x^{(0,p)}$ on a majority of the benchmark problems, while the other solvers do so for over 95% of the problems. For the more restrictive τ, CMS and DMS perform nearly identically, while SMS and GDMS are shown to be even more robust. GYY is relatively faster at finding small Ψ values in the initial $150(n+1)$ function evaluations. Note that the data profiles for SMS are improved when moving from function value measures (Figure 5.3) to stationarity measures (Figure 5.4). We attribute this behavior to the fact that some stochastic instances find

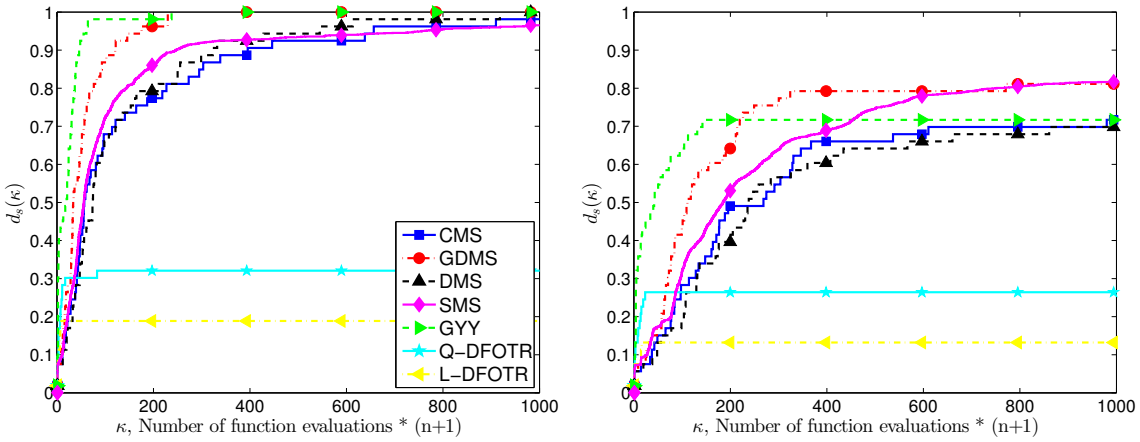

Figure 5.4: Data profiles based on the Ψ convergence measure (5.28) for $\tau = 10^{-3}$ (left) and $\tau = 10^{-7}$ (right).

stationary points with relatively worse function values (recall Figure 5.2).

Before proceeding, we note that although these tests show that GDMS is efficient when evaluations are performed sequentially, the other variants have the ability to utilize $n + 1$ evaluations concurrently and therefore might prove more useful in a parallel setting.

5.6.5 Comparison with Gradient Sampling

We also compare the performance between a variant of manifold sampling that uses some gradient information (SMS-G) and GRAD-SAMP, a MATLAB implementation of gradient sampling from [11] that uses gradient information at every evaluated point. This code was run with its default settings and a budget of $1000(n + 1)$ Jacobian (and hence gradient) evaluations. Since GRAD-SAMP does not proceed from a nondifferentiable initial point, the five problems with nondifferentiable starting points were perturbed by machine epsilon. We also extended the set of sampling radii in GRAD-SAMP from $\{10^{-4}, 10^{-5}, 10^{-6}\}$ to $\{10^{-4}, 10^{-5}, \ldots, 10^{-16}\}$ to avoid early termination.

As suggested by its name, SMS-G is SMS from Section 5.6.1 with the following modifications. The model building step, Line 4 in Algorithm 2, directly uses the Jacobian $\nabla F(x^k)$ and thus $M(x) = F(x^k) + \nabla F(x^k)(x - x^k)$. Since the default settings in GRAD-SAMP samples $\min(n + 10, 2n)$ gradients per iteration, SMS-G has this manifold sampling rate as opposed to the n points sampled each iteration by SMS.

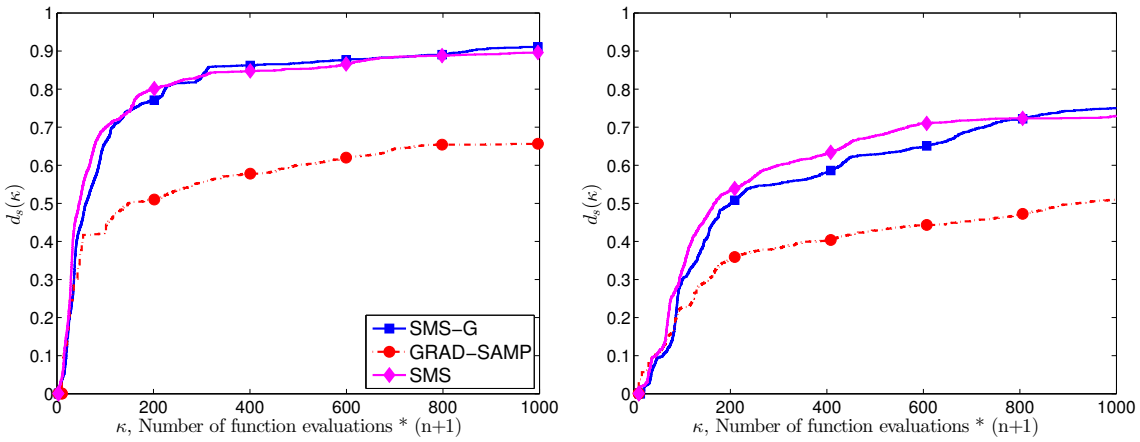

Figure 5.5: Data profiles based on the function value convergence measure (5.27) for $\tau = 10^{-3}$ (left) and $\tau = 10^{-7}$ (right). Note that SMS samples n points per iteration while SMS-G and GRAD-SAMP sample $\min\{n+10, 2n\}$ points per iteration.

Notice that because SMS-G computes a new Jacobian only immediately following a successful iteration, it incurs at most one Jacobian evaluation per iteration. The manifold sampling step and the evaluation of the trial point in SMS-G require only function evaluations (i.e., not Jacobian evaluations), and so SMS-G incurs at most $\min(n+11, 2n+1)$ function evaluations per iteration. On the other hand, GRAD-SAMP can require a bundle of $\min(n+11, 2n+1)$ gradient (and hence Jacobian) and corresponding function evaluations per iteration. Thus, in our data profiles measured in terms of function evaluations, every function evaluation used by GRAD-SAMP entails a Jacobian evaluation; SMS-G function evaluations include a Jacobian only evaluation for a fairly small (always less than 20%) proportion of the function evaluations. That is, each function evaluation within gradient sampling includes a "free" Jacobian evaluation that is not accounted for in the presented data profiles. SMS-G uses a "free" Jacobian evaluation on fewer than 20% of the function evaluations.

Data profiles are shown in Figure 5.5 and Figure 5.6 for an experiment where 30 stochastic runs were performed for both solvers. We also compare results with 30 instances of the Jacobian-free SMS described in Section 5.6.1. The gradient sampling method performs significantly worse than either manifold sampling method. Furthermore, the similar performance exhibited by the Jacobian-based SMS-G and the Jacobian-free SMS indicates that

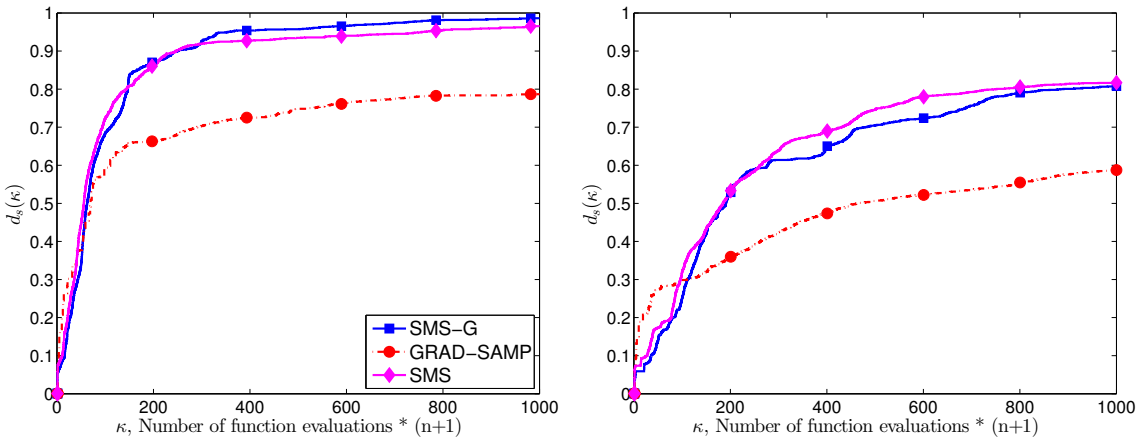

Figure 5.6: Data profiles based on the Ψ convergence measure (5.28) for $\tau = 10^{-3}$ (left) and $\tau = 10^{-7}$ (right).

the performance of SMS-G would likely further improve if the sampling rate were reduced.

5.7 Discussion

The driving force behind the proposed manifold sampling algorithm is that search directions are computed by using a finitely generated set,

$$\mathbf{co}\left\{\nabla M(x^k)\mathbf{sign}(y^j) + \sum_{i:[\mathbf{sign}(y^j)]_i=0} t_i \nabla m^{F_i}(x^k) \;:\; t \in \{-1,0,1\}^r, y^j \in Y(x^k;\Delta_k)\right\},$$

which differs from the finitely generated set used by gradient sampling,

$$\mathbf{co}\left\{\nabla M(y^j)\mathbf{sign}(y^j) \;:\; y^j \in \mathcal{D} \cap Y(x^k;\Delta_k)\right\}.$$

Our tests on ℓ_1 functions show that the manifold sampling strategies compare favorably with a gradient sampling approach.

Our presentation in Sections 5.3 and 5.4 focused on the case of (5.2) when $f_s = 0$. Since the presence of a nontrivial f_s does not affect the manifolds of f, such f_s can naturally be addressed by a shift of the generator set to $\nabla f_s(x^k)+\mathfrak{G}_k$, inclusion of f_s in the master model (e.g., $\nabla m^f(x^k+s) = \left\langle s, \mathrm{Proj}\left(0, \nabla f_s(x^k) + \nabla M\left(x^k\right)\partial h\left(F\left(x^k\right)\right)\right)\right\rangle$), and an analogous to ρ_k.

Furthermore, although the present work targets composite problems (5.2) for the par-

ticular nonsmooth function $h(u) = \|u\|_1$, the approach can be extended to other functions h, provided one can classify points into smooth manifolds (either with zeroth-order information or inexact first-order information). In the case of ℓ_1 functions, this classification was determined by the trivial evaluation of the sign pattern of $F(y)$ for a sample $y \in Y(x, \Delta)$. In the case of minimax objective functions of the form

$$h(u) = \max_{i=1,\ldots,r} u_i \qquad \text{or} \qquad \max_{i=1,\ldots,r} |u_i|,$$

this classification is determined by what Hare and Nutini refer to as the "active set" at a point [39]. In general, in any setting where the form of the subdifferential ∂h at any point is known, a setting that subsumes much of the work in the nonsmooth composite optimization literature, an analogous version of Algorithm 2 can be proposed.

In particular, the manifold sampling approach does not rely on convexity of the function h. This is in contrast to methods that solve the nonsmooth subproblem (5.9). An example of such a method is the GYY code modified from [36], which we showed can slightly outperform manifold sampling on ℓ_1 problems.

We are also interested in efficient and greedy updates of sample sets for manifolds and/or models in both settings where function (and Jacobian) evaluations are performed sequentially and concurrently. Our implementation of GDMS is a first step in this direction. Furthermore, natural questions arise about the tradeoff between the richness of manifold information required to reach early termination of the inner while loop in Algorithm 2 and the efficiency to guarantee that the sample set is, for example, well poised for model building.

Chapter 6

HAILSTORM for Non-convex Empirical Risk Minimization

6.1 Non-convex Empirical Risk Minimization

In this short chapter, we will consider the well-known task of empirical risk minimization for binary (linear) classification. That is, we assume that data is generated from some unknown distribution of features and binary labels on $X \times \{-1, 1\}$, where $X \subseteq \mathbb{R}^d$, and we seek a *classifier* $(w, w_0) \in \mathbb{R}^d \times \mathbb{R}$ such that

$$\mathbb{E}_{(x,y)\sim X\times\{-1,1\}}\mathbb{I}[\mathbf{sign}(w^\top x + w_0) = y]$$

is maximized, where $\mathbb{I}[\cdot]$ denotes an indicator function. To simplify notation, we will suppose that the d-th feature of each datapoint x^i is the constant 1, and thus the bias term w_0 is simply part of the classifier vector w. For empirical risk minimization, we then assume we are given a *training set* of pairs $\{(x^i, y^i)\}_{i=1}^n \sim X \times \{-1, 1\}$

Based solely on this description of the binary classification task, one expects that, assuming the training set is indeed a representative sample of the distribution on $X \times \{-1, 1\}$, the classifier w obtained by solving the following optimization problem will generalize and maximize predictive ability on unseen datapoints coming from $X \times \{-1, 1\}$:

$$\min_w \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[\text{sign}(y^i(w^\top x^i)) = -1]. \tag{6.1}$$

However, because the indicator function $\mathbb{I}[\cdot]$ is discontinuous, and moreover has zero gradient everywhere its gradient is defined, methods of continuous optimization cannot optimize (6.1) directly. It has been shown (see e.g. [7, 69]) that the solution of (6.1) is an NP-hard problem. Therefore, a majority of successful applications of optimization to machine learning use a *convex approximation* of (6.1) such as the smooth logistic loss of logistic regression,

$$\min_w \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + \exp\left(-y^i(w^\top x^i)\right)\right), \tag{6.2}$$

or the nonsmooth hinge loss used in support vector machines (SVMs). This is largely motivated by the fact that globally optimal solutions to approximations like (6.2) can be obtained efficiently. There also exist certain statistical guarantees [5] regarding the quality of classifiers obtained from solving convex problems such as (6.2) in terms of their performance on the actual problem (6.1).

Despite these positive results, a good deal of work (see in particular [25, 53]) has exhibited that these convex approximations suffer from a lack of robustness in the presence of misclassification error, i.e. generalization severely degrades in instances where the labels y^i in the training set are incorrect or "flipped", as we will demonstrate through numerical experiments in this chapter.

A natural alternative to consider, then, is the use of *non-convex* loss functions in empirical risk minimization. In particular, we will consider the following version of a well-known non-convex (but continuous and smooth) problem approximating (6.1), which we will refer to throughout this chapter as *sigmoidal loss*:

$$\min_w \ell^K(w) \triangleq \frac{1}{n} \sum_{i=1}^{n} \left[\frac{2\exp[-Ky^i(w^\top x^i)]}{1 + \exp[-Ky^i(w^\top x^i)]} - 1\right], \tag{6.3}$$

parameterized by K, a control on the smoothness of the approximation of (6.1). See Figure 6.1.

Note that the range of the objective in (6.3) has been rescaled to $(-1, 1)$. For an

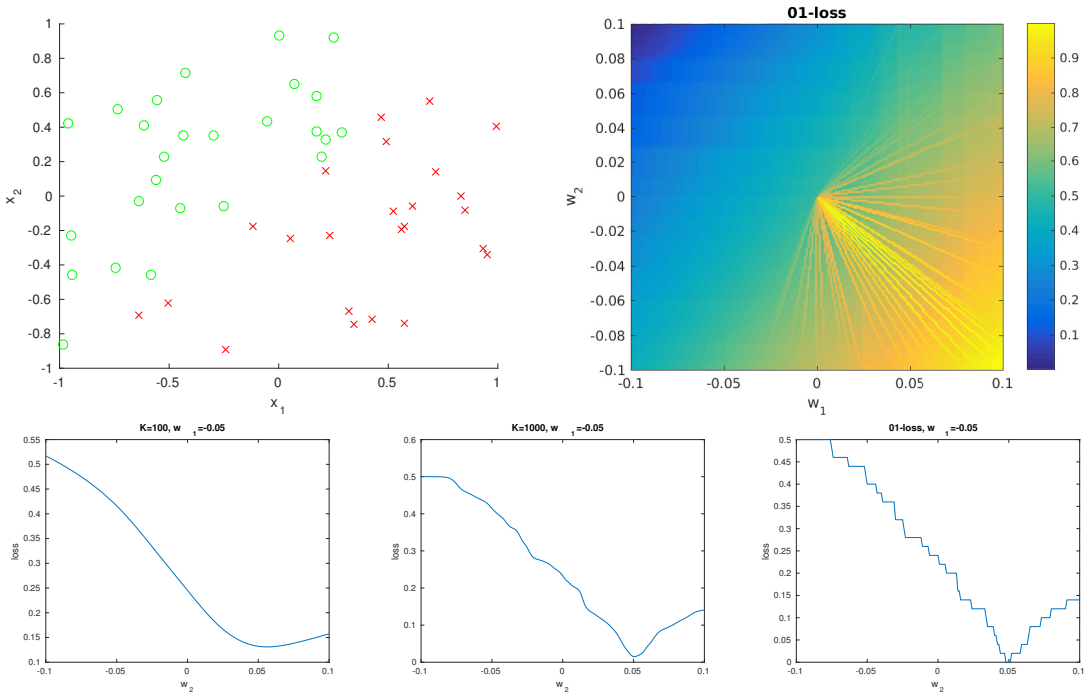

Figure 6.1: **(Top left):** A synthetic random two-dimensional dataset. Green circles have positive labels and red crosses have negative labels. Datapoints satisfying $x_1 - x_2 > 0$ are labelled negative, otherwise they are labelled positive. **(Top right):** A contour plot of the value of the indicator loss with respect to the synthetic data. Observe that plotting artifacts obscure the fact that the contour plot is constant along any given ray, and so any vector in the direction $[-0.05; 0.05]$ is optimal with no loss. Observe also that if the constant function value along a given ray is θ, then the function value along the negative of said ray is $1 - \theta$. **(Bottom row):** Slices of the contour plot when w_1 is fixed to -0.05 for $K = 100$, $K = 1000$, and "$K = \infty$" in the sigmoidal loss from (6.3)

explanation of practical reasons (in the context of neural networks) to use such a loss function as opposed to a loss function with range in $[0, 1]$, see [50].

As is standard practice in machine learning, we may want to add some form of regularization to the problem given in (6.3) to prevent overfitting. The standard choices of regularized problems use ℓ_1-regularization

$$\min_w \ell^{K,1}(w) \triangleq \frac{1}{n} \sum_{i=1}^{n} \left[\frac{2 \exp[-K y^i (w^\top x^i)]}{1 + \exp[-K y^i (w^\top x^i)]} - 1 \right] + \frac{\lambda}{2} \|w\|_1 \tag{6.4}$$

or ℓ_2-regularization

$$\min_w \ell^{K,2}(w) \triangleq \frac{1}{n} \sum_{i=1}^{n} \left[\frac{2 \exp[-K y^i (w^\top x^i)]}{1 + \exp[-K y^i (w^\top x^i)]} - 1 \right] + \frac{\lambda}{2} \|w\|_2^2, \tag{6.5}$$

where in both (6.4) and (6.5), $\lambda > 0$ is a constant.

As can be observed in Figure 6.1, however, as $K \to \infty$, ℓ_2 regularization (6.5) becomes meaningless since (6.1) is scale-invariant with respect to w. Thus, we will focus our attention on the ℓ_1-regularized problem (6.4). However, (6.4) is a nonsmooth problem and so a direct of application of the STORM algorithm should not work. Thus, we propose a hybrid of STORM and manifold sampling, which we call HAILSTORM (Hierarchical Approximations to Indicator Loss via STORM), to solve the special case of problem (6.4).

6.2 HAILSTORM

As is necessary for any STORM algorithm, we first describe a means of constructing models and obtaining function value estimates. We will not prove that the model construction and function value estimation schemes proposed in this section do, in fact, yield α-probabilistically fully-linear models or β-probabilistically accurate estimates, respectively, but we will indicate that similar constructions have been considered previously in the literature when appropriate.

To obtain stochastic estimates of the sigmoidal loss given in (6.3), given a sample of indices $S \subseteq \{1, 2, \ldots, n\}$, we define

$$\ell_S^K(w) \triangleq \frac{1}{|S|} \sum_{i \in S} \left[\frac{2 \exp[-K y^i (w^\top x^i)]}{1 + \exp[-K y^i (w^\top x^i)]} - 1 \right]. \tag{6.6}$$

6.2.1 Function Value Estimates (Adaptive Sampling)

To obtain function value estimates, we propose the use of *adaptive sampling*; i.e. given a classifier w to evaluate, we will randomly sample without replacement $S \subseteq \{1, 2, \ldots, n\}$ large enough such that the standard error of $\ell_S^K(w)$ is (approximately) decreased to $\mathcal{O}(\delta^2)$, in line with the definition of fully-linearity. In particular, given an algorithmic parameter $\kappa_f > 0$ and an algorithmically-determined radius δ, we want to satisfy the following condition on standard error:

$$\frac{\hat{\sigma}(w, S)}{\sqrt{|S|}} \leq \kappa_f \delta^2, \tag{6.7}$$

where $\hat{\sigma}(w, S)$ denotes the standard deviation

$$\hat{\sigma}(w, S) \triangleq \sqrt{\sum_{i=1}^{|S|} \frac{(\ell_i^K(w) - \ell_S^K(w))^2}{|S|}}.$$

Although one could consider an algorithm for computing function value estimates where one starts with an initial sample S_0 and gradually augments the sample until the condition (6.7) is satisfied (which is essentially what is proposed for Monte Carlo sampling in the ASTRO-DF framework [68]), this may be costly in terms of time, since computing the standard error as side information incurs an additional cost over simple function evaluation. Thus, following the same basic idea of dynamic sample sizes as in [13], we propose defining within the STORM algorithm a nondecreasing variable sample size S_k to be used in the k-th iteration to obtain the estimates f_k^0 and f_k^s necessary in the general STORM framework in Algorithm 1. That is, having obtained the standard error $\hat{\sigma}(w_k, S_{k-1}) \triangleq \hat{\sigma}_{k-1}$ in the $(k-1)$-th iteration as a byproduct of computing the estimate f_{k-1}^s or f_k^0 in successful or unsuccessful iterations respectively, we draw in the k-th iteration a new set of random samples S_k satisfying

$$|S_k| = \min\left\{n, \max\left\{\left\lceil \frac{\hat{\sigma}_{k-1}^2}{\kappa_f^2 \delta_k^4} \right\rceil, |S_{k-1}|, k\right\}\right\} \tag{6.8}$$

for the computations of f_k^0 and f_k^s. The insistence in (6.8) that $|S_k|$ grow at least as fast as k is a matter of practicality.

6.2.2 Random Models (Mini-Batch Stochastic Gradients)

In terms of computing random models intended to be probabilistically fully-linear, we follow a similar method of dynamic sample sizes for computing mini-batch stochastic gradients, which is more directly in line with [13]. That is, given a classifier w and a trust region parameter δ, we attempt to ensure that a sufficiently large sample S is selected so that the standard error of the gradient $\nabla \ell_S^K(w)$ is in $\mathcal{O}(\delta)$. That is, given a parameter $\kappa_g > 0$, we

seek a sample S such that

$$\sqrt{\frac{\|\mathrm{Var}(w,S)\|_1}{|S|}} \leq \kappa_g \delta, \tag{6.9}$$

where the vector-valued operator Var is defined as

$$\mathrm{Var}(w,S) = \frac{1}{|S|-1}\sum_{i\in S}(\nabla \ell_S^K(w) - \nabla \ell_i^K(w))^2,$$

with the square taken component-wise. Using the same sort of estimation that led to the sample size prescription (6.8) for computing function value estimates, the size of the mini-batch stochastic gradient in the k-th iteration is chosen to satisfy

$$|S_k| = \min\left\{n, \max\left\{\left\lceil \frac{\|\mathrm{Var}(w,S_{k-1})\|_1}{\kappa_g^2\delta^2}\right\rceil, |S_{k-1}|, k\right\}\right\}. \tag{6.10}$$

6.2.3 Manifold Sampling to Handle Nonsmoothness

In most adaptations of stochastic gradient methods to handle the nonsmoothness caused by an ℓ_1-regularization term, proximal operators [65] are used, which in the ℓ_1 case reduces to so-called *soft-thresholding*, see e.g. [6]. Here, since STORM uses a trust-region framework, we can make use of a greatly-simplified variant of the manifold sampling algorithm described in Chapter 5.

Let $h(\cdot) = \frac{\lambda}{2}\|\cdot\|_1$. In a trust-region framework, on the kth iteration, we can easily determine a superset of the smooth manifolds induced by h that intersect the current trust-region $\mathcal{B}(w_k, \delta_k)$. Consider the *active* indices $\mathcal{J}^* = \{j : |(w_k)_j| < \delta_k\}$. Additionally define $\mathcal{J}^+ = \{j : (w_k)_j \geq \delta_k\}$ and $\mathcal{J}^- = \{j : (w_k)_j \leq -\delta_k\}$. Then, letting $g = \nabla \ell^K(w_k)$, where $\ell^K(\cdot)$ is the sigmoidal loss in (6.3), the *master model* gradient of $\ell^{K,1}(w_k)$ with respect to the active manifolds represented in $\mathcal{J}^*$ is given as $g + y^*$ where y^* is a solution to

$$\begin{array}{lll} \min\limits_{y \in \mathbb{R}^d} & (g+y)^\top (g+y) & \\ \text{s. to} & y_j = -\lambda & \forall j \in \mathcal{J}^- \\ & y_j = \lambda & \forall j \in \mathcal{J}^+ \\ & -\lambda \leq y_j \leq \lambda & \forall j \in \mathcal{J}^* \end{array} \tag{6.11}$$

Observe that (6.11) is a convex bound-constrained quadratic optimization problem, a well-studied problem for which fast solvers exist in practice. Since for any w, every generator of the subdifferential $\partial h(w)$ has all of its entries as $\pm\lambda$, we see that (6.11) is just a specialized case of the minimum-norm element problem cast in (5.11). Thus, in our proposed algorithm, we will solve one quadratic optimization problem of the form (6.11) per iteration to obtain y^*, and simply replace the (stochastic) gradient $\nabla \ell^K(w_k)$ ($\nabla \ell^K_{S_k}$ in the stochastic case) with $\nabla \ell^K(w_k) + y^*$ (respectively, $\nabla \ell^K_{S_k}(w_k) + y^*$).

In terms of a ratio test for step acceptance, as in the ratio test used for manifold sampling (5.14), we will insist that decrease in the objective be made from the perspective of the manifold of the trial step $w_k + s_k$. That is, we define

$$\rho_k \triangleq \frac{\ell^K_{S_k}(w_k) - \ell^K_{S_k}(w_k + s_k) + \dfrac{\lambda}{2}(\langle w_k, \operatorname{sign}(w_k + s_k)\rangle - \|w_k + s_k\|_1)}{m_k(w_k) - m_k(w_k + s_k) + \dfrac{\lambda}{2}(\langle w_k, \operatorname{sign}(w_k + s_k)\rangle - \|w_k + s_k\|_1)}, \tag{6.12}$$

where $m_k(\cdot) = \langle \nabla \ell^K_{S_k}(w_k) + y^*, \cdot \rangle$. Once again, this is simply a specialized version of the ρ_k presented in Algorithm 2, since given w we can define the jth "component model" as w_j for $j = 1, 2, \ldots, d$, and this "model" choice yields zero approximation error.

We recall that Algorithm 2 from Chapter 5 was for deterministic nonsmooth functions, and no results were proven about variants of that method where the component functions were stochastic. We believe that because Algorithm 2 iteratively considers smoothed objectives in a derivative-free trust-region framework, the analysis of STORM in Chapter 2 could likely be adapted and applied to a stochastic variant of Algorithm 2 under similar assumptions on stochasticity of the component functions. Seeing if the analysis truly carries over in such a way is a subject for future work.

Algorithm 5: HAILSTORM$(w_0, \delta_0, K, \lambda)$

(Initialization): Choose $\gamma > 1$, $\eta_1 \in (0,1)$, $\eta_2 > 0$, $\epsilon_F > 0$, $\kappa_f > 0$, $\kappa_g > 0$, $\epsilon > 0$, $\delta_{\max} \geq \delta_0$.
$k \leftarrow 0$.
(Model construction): Compute a stochastic gradient $\nabla \ell^K_{S_k}(w_k)$ such that S_k satisfies (6.10).
$g_k \leftarrow \nabla \ell^K_{S_k}(w_k) + y^*$, where y^* is the optimal solution to (6.11).
(Step calculation): $s_k \leftarrow -\delta_k g_k / \|g_k\|$.
(Estimates calculation): Obtain independent estimates $\ell^K_{S_k}(w_k)$ and $\ell^K_{S_k}(w_k + s_k)$ so that S_k satisfies (6.8) for the respective estimates of $\ell^K(w_k)$ and $\ell^K(w_k + s_k)$.
(Acceptance of the trial point): Compute ρ_k as in (6.12).
if $\rho_k \geq \eta_1$ and $\|g_k\| \geq \eta_2 \delta_k$ **then**
  $w_{k+1} \leftarrow w_k + s_k$
else
  $w_{k+1} \leftarrow w_k$.
end if
(Trust-region radius update):
if $\rho_k \geq \eta_1$ and $\|\nabla \ell^K_{S_k}(w_k)\| \geq \eta_2 \delta_k$ **then**
  $\delta_{k+1} \leftarrow \min\{\gamma \delta_k, \delta_{\max}\}$
else
  $\delta_{k+1} = \gamma^{-1} \delta_k$.
end if
(Iterate) $k \leftarrow k + 1$, go to step 1.

We present a statement of HAILSTORM in Algorithm 5.

6.3 Computational Experiments

6.3.1 Comparison with Stochastic Gradient Descent Methods

We first demonstrate that HAILSTORM finds high-quality local minima to the sigmoidal loss objective. We will illustrate this with two well-known datasets from LIBSVM [14], namely the adult (a9a) and the web (w8a) dataset. These datasets are pre-split into a training and testing dataset. We compare the performance of HAILSTORM with a stochastic mini-batch gradient descent (SGD) method and a stochastic reduced variance gradient (SVRG) method. It has been shown in independent papers [1, 61] that SVRG with a fixed step-size converges to first-order stationary points of a non-convex objective under mild assumptions.

The mini-batch size for both of these methods and both datasets was set to $\lceil n^{2/3} \rceil$. We used a decaying step-size sequence $\eta_0(1+\eta_1 k)^{-1}$, where k is the number of iterations, and we tuned the parameters on the training set by selecting the value of $\eta_0 \in [10^{-7}, 3 \cdot 10^{-7}, 10^{-6}, 3 \cdot 10^{-6}, \ldots, 1, 3, 10]$ and the value of $\eta_1 \in [0, 1/3, 1/2, 1]$ that led to the best final value of the training loss after 50 data passes. Meanwhile, for HAILSTORM, we simply implemented Algorithm 5 in Matlab with fixed parameter choices $\gamma = 2, \eta_1 = 10^{-3}, \eta_2 = 10^{-3}, \kappa_f = 10^3, \kappa_g = 10^3, \epsilon = 10^{-8}, \delta_{\max} = 10$, and $\delta_0 = 1$. All three algorithms (SGD, SVRG, and HAILSTORM) are given as an initial point w_0, an approximate global minimizer to the deterministic logistic loss problem (6.2) with the same regularization parameter λ as the problem we intend to solve, namely (6.5) with $K = 1$.

In this experiment, we run HAILSTORM exactly once with a single random seed. Thus, this very simple experiment demonstrates that with no parameter tuning whatsoever (and hence at a significantly lower overall cost), HAILSTORM tends to find as high quality solutions as the best-tuned versions of SGD or SVRG. We show in Figures 6.2 and 6.3 the performance of all three algorithms over 50 effective data passes. For SGD and SVRG, an effective data pass is counted as the number of times a single data point was accessed divided by the number of points n in the dataset (and hence represents the same effort as computing a full gradient). To yield a fair comparison, in HAILSTORM we count every access to a datapoint as a single access regardless of whether the access is being used to compute a function value or a gradient value. Arguably, this comparison is in fact biased against HAILSTORM, since the overhead of computing a function value is strictly less than the overhead needed to compute a gradient value.

The observed behavior in Figures 6.2 and 6.3 is likely due to the fact that STORM (and hence, HAILSTORM) is "function-value aware"; while it is theoretically guaranteed to find first-order stationary points provided the sequence of models and estimates are α-probabilistically fully linear and β-probabilistically accurate respectively, it is more inclined than the "function value-blind" SGD and SVRG methods to escape a local solution of poor objective quality. Comparatively, we see that while a well-tuned run of SVRG has a fairly stable trajectory in decreasing gradient norm, as is predicted by its theory, it gets

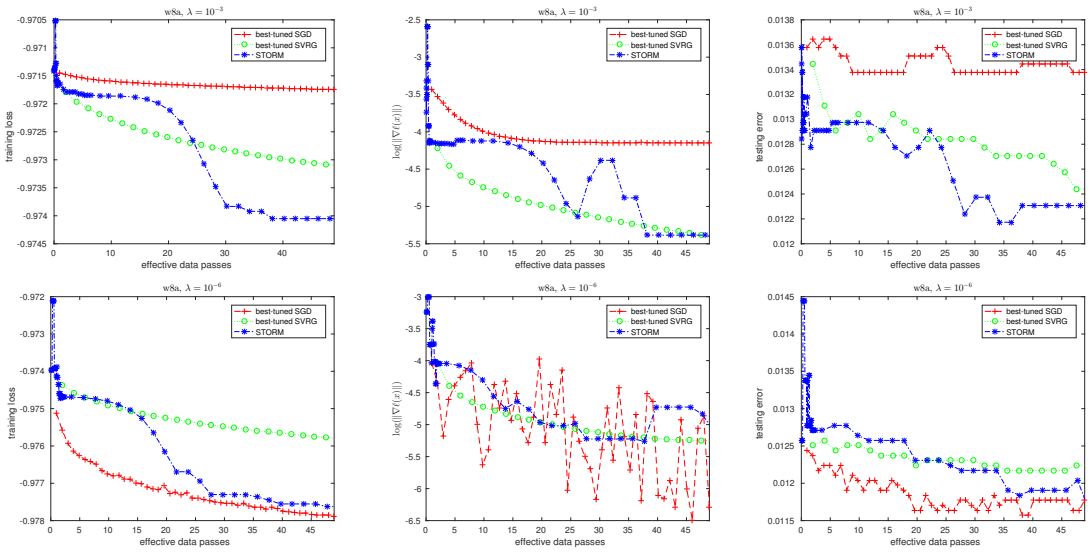

Figure 6.2: Results of the discussed experiment for ℓ_2-regularization parameter $\lambda = 10^{-3}$ **(top row)** and $\lambda = 10^{-6}$ **(bottom row)** on the w8a dataset . The left image shows the trajectory of training loss, the middle image shows the trajectory of the loss function gradient norm $\|\nabla\ell(w)\|$, and the right image shows the trajectory of the holdout testing error.

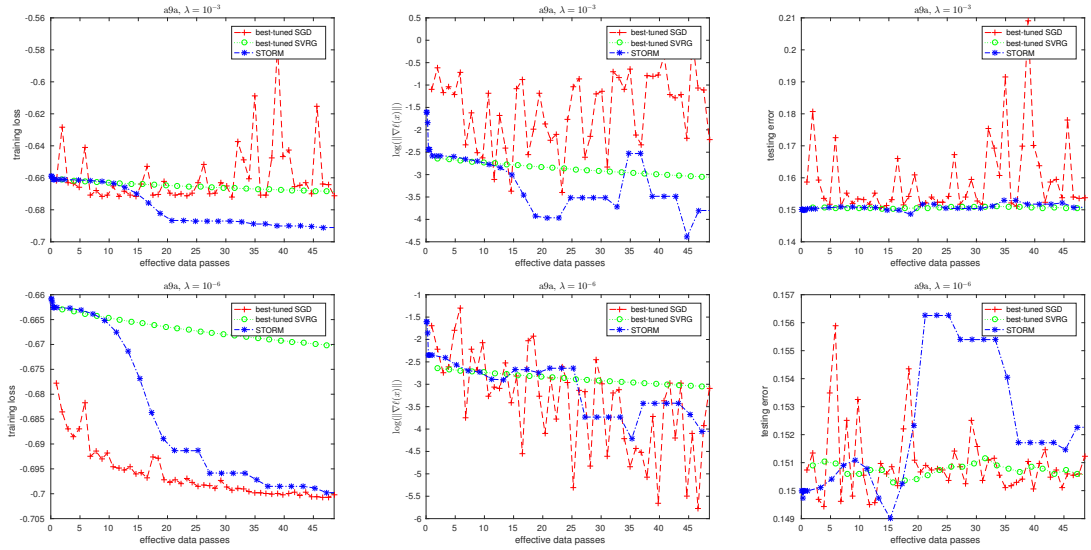

Figure 6.3: Same as in Figure 6.2, but on the a9a dataset.

stuck in poor-quality minima. On the other hand, and as is often observed in practice, a well-tuned run of SGD tends to find high-quality minima, although with wild gradient norm trajectories. Our implementation of HAILSTORM appears to be a good negotation between these two extremes. We reiterate that we only ran HAILSTORM once with a single random seed in each of these illustrations, and yet it consistently exhibits relatively

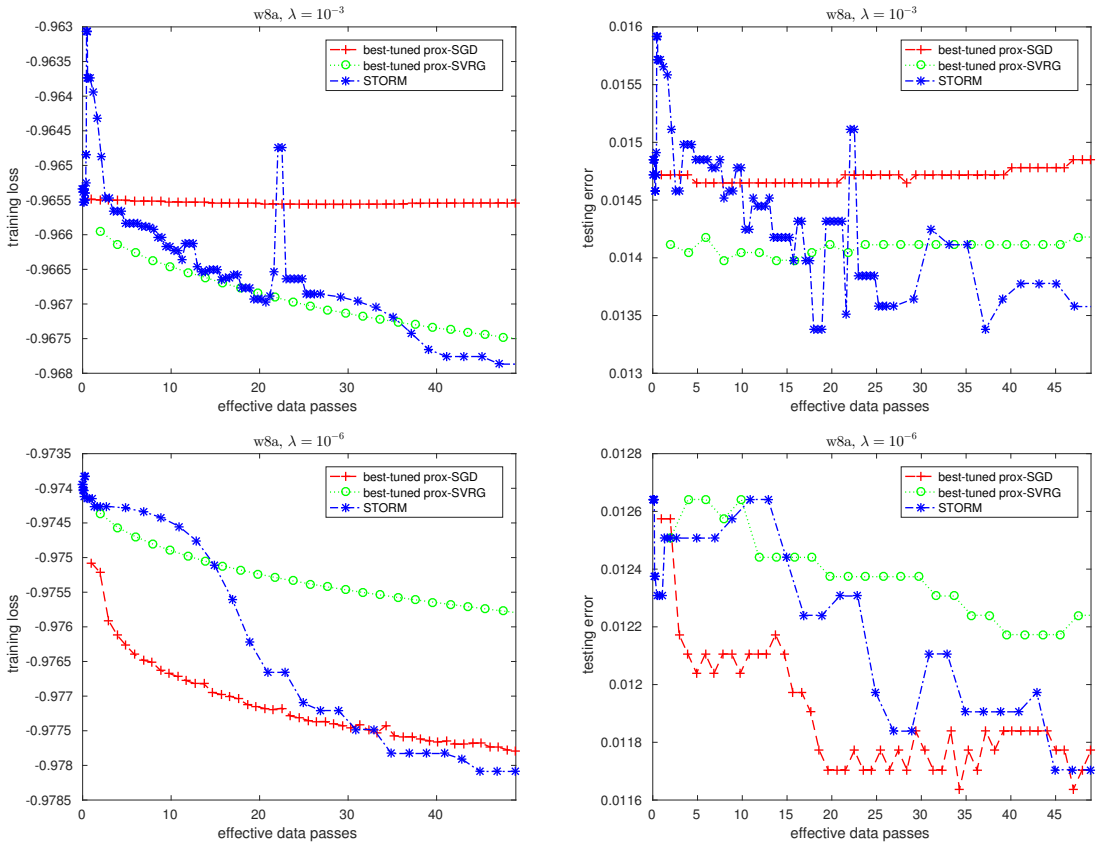

Figure 6.4: Results of the discussed experiment for ℓ_1-regularization parameter $\lambda = 10^{-3}$ **(top row)** and $\lambda = 10^{-6}$ **(bottom row)** on the w8a dataset . The left image shows the trajectory of training loss and the right image shows the trajectory of the holdout testing error.

good performance in all the criteria of training loss, gradient norm, and testing error.

We also make comparisons between our nonsmooth Algorithm 5 (i.e. $\lambda > 0$ in (6.4), proximal stochastic mini-batch gradient descent (prox-SGD), and proximal SVRG (prox-SVRG). For a good summary of convergence results surrounding prox-SGD and prox-SVRG, see [62]. We use the same parameters, tuning methods (or lack thereof in HAIL-STORM's case), and initializations as in the previous experiment. We illustrate the results in Figures 6.4 and 6.5. We omit a comparison of "gradient norms" since the notion of sub-gradient inherently differs between a proximal operator method and a manifold sampling method (which deliberately overestimates the subdifferential).

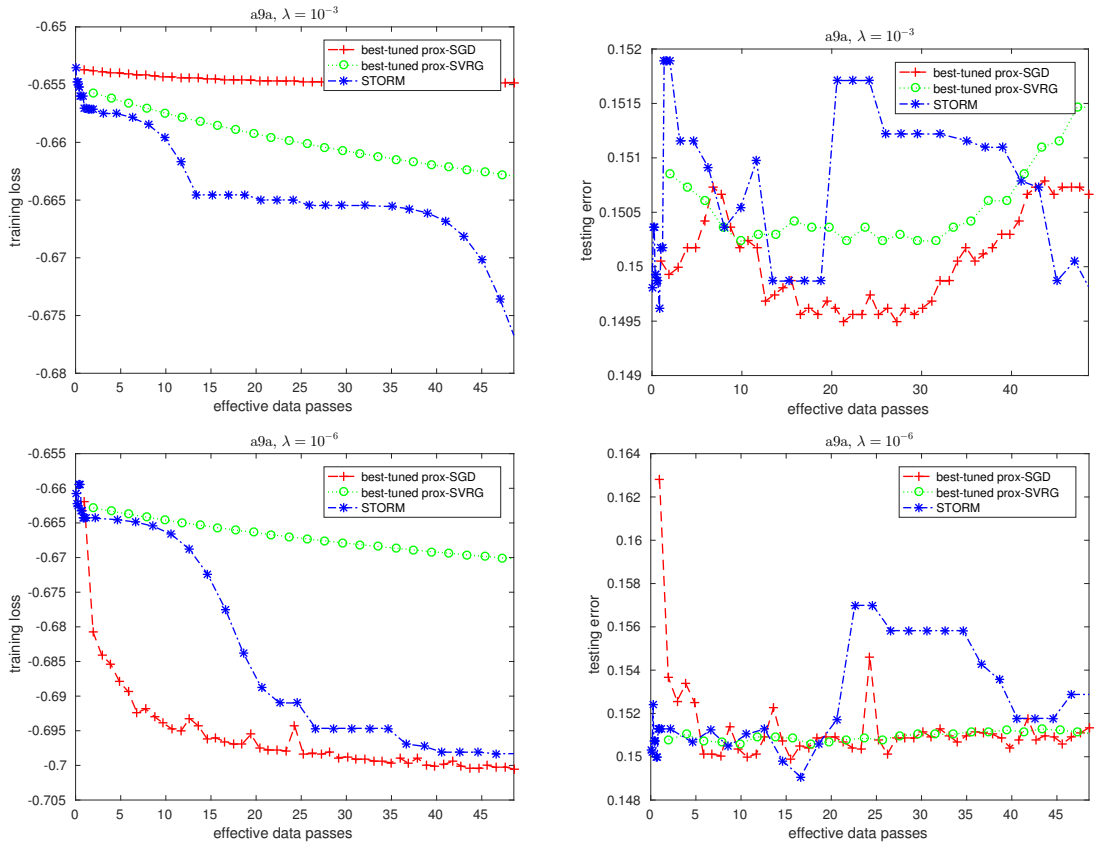

Figure 6.5: Same as in Figure 6.4, but on the a9a dataset.

6.3.2 Sigmoidal Loss vs. Logistic Loss

We now demonstrate the benefits of using the solutions to (6.4) obtained by HAILSTORM after a reasonable number of data passes, as opposed to using the globally optimal solution to a deterministic (ℓ_1-regularized) logistic loss problem

$$\min_w \frac{1}{n}\sum_{i=1}^{n} \log\left(1+\exp\left(-y^i(w^\top x^i)\right)\right) + \frac{\lambda}{2}\|w\|_1. \tag{6.13}$$

As motivated in Section 6.1, we are particularly interested in the robustness of learned classifiers in the presence of misclassification errors. Towards this end, we consider, in addition to the a9a and w8a datasets, 8 well-known datasets from the UCI Machine Learning Repository [51]. A summary description of the datasets used is given in Table 6.1. For the a9a and w8a datasets, to avoid confusion, we only made use of the training set in our experimental design, ignoring the given testing set.

	a9a	breast-cancer	ionosphere	magic	pima	satimage	segment	statlog-heart	statlog-shuttle	w8a
n (# of samples)	32561	683	351	19020	768	4435	2100	270	43500	49749
d (dimension)	123	9	14	10	8	36	19	13	9	300

Table 6.1: Problem data for UCI/LIBSVM datasets.

For a single run of the experiment we do the following for each dataset.

1. Randomly split the dataset into a training set T_{tr} and testing set T_{te} of size $\lceil 0.8n \rceil$ and $\lfloor 0.2n \rfloor$ respectively.

2. Contaminate the training set T_{tr} by choosing a fraction $p \in [0, 1]$ and multiplying the $\{-1, 1\}$-label of a randomly selected $\lceil p|T_{tr}| \rceil$ many points in T_{tr} by -1, i.e. by "flipping" the label of a fraction p of the training set.

3. Further randomly split the (already contaminated) training set T_{tr} into a training set T_{tr2} and validation set T_{val} of sizes $\lceil 0.8|T_{tr}| \rceil$ and $\lfloor 0.2|T_{tr}| \rfloor$ respectively.

4. For different values of an ℓ_1-regularizer $\lambda \in \{0, 1/n, 1/d, 1\}$, train a classifier using T_{tr2} by solving (6.13) to (approximate) global optimality.

5. Identify the value of λ from Step 4 that yields a classifier w^*_{log} with the highest accuracy on T_{val}. Denote this value of the regularization parameter by λ^*, and record the accuracy of w^*_{log} on T_{te}.

6. For $K \in \{1, 2, 10, 20, 100, 200\}$, run $HAILSTORM(w^*_{log}, 1, K, \lambda^*)$ using T_{tr2}.

7. Identify the value of K from Step 6 that yields a classifier w^*_{sig} with the highest accuracy on T_{val}, and record the error made by w^*_{sig} on T_e.

Thus, this experiment is intended to demonstrate that by using a reasonable classifier (i.e. one learned by optimizing logistic loss (6.13)) as an initial point, running HAILSTORM on a regularized sigmoidal loss problem (6.4) with a well-chosen value of the smoothing parameter K can improve the generalization error of that classifier. A summary of results of running this experiment 40 times with fraction of flips $p = 0, 0.125, 0.25$

	no flips		1/8 flips		1/4 flips	
	logistic	sigmoidal	logistic	sigmoidal	logistic	sigmoidal
a9a	$\mathbf{15.28 \pm 0.01}$	15.34 ± 0.01	$\mathbf{15.50 \pm 0.01}$	15.58 ± 0.01	$\mathbf{15.90 \pm 0.01}$	16.04 ± 0.01
breast-cancer	3.66 ± 0.04	$\mathbf{3.57 \pm 0.04}$	4.65 ± 0.04	$\mathbf{4.19 \pm 0.04}$	5.28 ± 0.06	$\mathbf{4.50 \pm 0.05}$
ionosphere	13.07 ± 0.12	13.04 ± 0.12	14.62 ± 0.14	$\mathbf{13.95 \pm 0.15}$	17.52 ± 0.18	$\mathbf{16.86 \pm 0.14}$
magic	20.98 ± 0.02	$\mathbf{20.37 \pm 0.02}$	21.32 ± 0.02	$\mathbf{20.48 \pm 0.02}$	21.65 ± 0.02	$\mathbf{20.69 \pm 0.02}$
pima	33.44 ± 0.13	$\mathbf{23.68 \pm 0.09}$	34.07 ± 0.11	$\mathbf{27.15 \pm 0.19}$	34.07 ± 0.11	$\mathbf{31.90 \pm 0.19}$
satimage	9.20 ± 0.02	$\mathbf{9.06 \pm 0.03}$	9.21 ± 0.02	$\mathbf{9.06 \pm 0.03}$	9.30 ± 0.02	$\mathbf{9.06 \pm 0.03}$
segment	0.23 ± 0.01	0.23 ± 0.01	1.33 ± 0.02	$\mathbf{0.59 \pm 0.01}$	2.35 ± 0.03	$\mathbf{1.37 \pm 0.02}$
statlog-heart	15.62 ± 0.19	$\mathbf{15.12 \pm 0.17}$	17.04 ± 0.20	$\mathbf{16.85 \pm 0.18}$	25.49 ± 0.37	$\mathbf{20.80 \pm 0.23}$
statlog-shuttle	3.51 ± 0.01	$\mathbf{2.36 \pm 0.00}$	8.03 ± 0.01	$\mathbf{2.48 \pm 0.00}$	8.92 ± 0.01	$\mathbf{2.52 \pm 0.00}$
w8a	1.44 ± 0.01	$\mathbf{1.28 \pm 0.00}$	1.49 ± 0.00	$\mathbf{1.47 \pm 0.00}$	1.66 ± 0.00	$\mathbf{1.64 \pm 0.00}$

Table 6.2: Test errors (in %) of classifiers trained by logistic and cross-validated sigmoidal loss with cross-validated ℓ_1 regularization. Each entry represents mean error $\pm$ one standard error.

	no flips		1/8 flips		1/4 flips	
	logistic	sigmoidal	logistic	sigmoidal	logistic	sigmoidal
a9a	15.32 ± 0.01	15.33 ± 0.01	$\mathbf{15.54 \pm 0.01}$	15.62 ± 0.01	$\mathbf{15.93 \pm 0.01}$	16.02 ± 0.02
breast-cancer	3.92 ± 0.06	$\mathbf{3.65 \pm 0.06}$	4.66 ± 0.06	$\mathbf{3.75 \pm 0.05}$	5.17 ± 0.08	$\mathbf{4.31 \pm 0.06}$
ionosphere	14.33 ± 0.14	14.33 ± 0.13	17.67 ± 0.14	$\mathbf{16.52 \pm 0.16}$	21.52 ± 0.19	$\mathbf{18.86 \pm 0.17}$
magic	20.90 ± 0.02	$\mathbf{20.28 \pm 0.02}$	21.24 ± 0.02	$\mathbf{20.38 \pm 0.02}$	21.58 ± 0.02	$\mathbf{20.70 \pm 0.02}$
pima	$\mathbf{23.05 \pm 0.10}$	23.40 ± 0.12	$\mathbf{24.97 \pm 0.12}$	26.06 ± 0.13	34.07 ± 0.11	$\mathbf{27.15 \pm 0.19}$
satimage	9.05 ± 0.03	9.05 ± 0.03	9.14 ± 0.03	9.12 ± 0.03	9.31 ± 0.03	$\mathbf{9.23 \pm 0.03}$
segment	0.27 ± 0.01	0.27 ± 0.01	1.53 ± 0.02	$\mathbf{0.63 \pm 0.02}$	2.52 ± 0.04	$\mathbf{1.09 \pm 0.02}$
statlog-heart	15.99 ± 0.16	15.99 ± 0.18	18.58 ± 0.19	$\mathbf{17.22 \pm 0.17}$	24.01 ± 0.29	$\mathbf{22.22 \pm 0.26}$
statlog-shuttle	3.47 ± 0.01	$\mathbf{2.36 \pm 0.01}$	7.99 ± 0.01	$\mathbf{2.51 \pm 0.01}$	8.90 ± 0.01	$\mathbf{2.58 \pm 0.01}$
w8a	1.49 ± 0.01	$\mathbf{1.33 \pm 0.00}$	1.49 ± 0.00	$\mathbf{1.46 \pm 0.00}$	1.67 ± 0.00	1.67 ± 0.00

Table 6.3: Test errors (in %) of classifiers trained by logistic and cross-validated sigmoidal loss with no regularization. Each entry represents mean error $\pm$ one standard error.

is given in Table 6.2. As a form of control, we also ran the experiment to see the effects of using no regularization at all, by effectively only considering $\lambda = 0$ in in Step 4, meaning that HAILSTORM is not making use of any of the manifold sampling machinery. The results of that experiment are shown in Table 6.3. In all experiments, when running HAILSTORM in Step 6 of the experiment, we used all the same algorithmic parameters as in Section 6.3.1.

We remark that in the ℓ_1-regularization experiment summarized in Table 6.2, with the exception of the a9a dataset, there is a consistent preference to use HAILSTORM as a post-processing method after minimizing logistic loss, particularly in noisier regimes of misclassification error. Comparing the results of using cross-validated ℓ_1-regularization to not using regularization is a bit more mixed, as is frequently observed to be the case in practice. For instance, just focusing on the very noisy regime (1/4 flips) and comparing the results of the sigmoidal loss with or without regularization, we see that using cross-

validation with regularization improves generalization error in a statistically significant way on the datasets ionosphere, satimage, statlog-heart, statlog-shuttle, and w8a, but actually hinders generalization error in a statistically significant way on breast-cancer, pima, and segment.

Chapter 7

Conclusion

In the first part of this thesis, Chapters 2-4, we extended the well-known framework of model-based trust-region methods for DFO to a new framework, STORM. In Chapter 2, we considered STORM as a method for minimization of a general unconstrained stochastic objective function f. We assumed that f was sufficiently smooth, but that any query of the function value $f(\cdot)$ would be contaminated by stochastic noise. In a break with the majority of stochastic optimization literature, our assumptions on the distribution of the noise were virtually nonexistent. Instead, we proved that provided one can generate what we called an α-probabilistically fully-linear sequence of models and a β-probabilistically ϵ_F-accurate sequence of function value estimates during the course of a STORM algorithm, then the iterates of the STORM algorithm converge almost surely to a first-order stationary point. This break from conventional assumptions on noise distributions allowed us to yield provably convergent algorithms even in situations where noise is occasionally dominating, where we explicitly quantify through α and β what is meant by dominating.

In Chapter 3, we then extended the analysis performed in Chapter 2 to prove expected global convergence rates of a STORM algorithm. In particular, recognizing that the iterates $\{X_k\}$ (and intermediate quantities, e.g. the trust-region radii $\{\Delta_k\}$) of STORM describe a stochastic process, we defined a stopping time T_ϵ as the number of iterates before the first time $\|\nabla f(X_k)\| \leq \epsilon$ is satisfied. We then proved that in expectation, $T_\epsilon \in \mathcal{O}(1/\epsilon^2)$, where the big-O contains an explicit (and intuitive) dependence on the parameters α and

β from the respective definitions of α-probabilistically fully-linear and β-probabilistically ϵ_F-accurate. This Chapter leads to some very interesting open questions. In particular, although we proved a bound on the expectation of T_ϵ, bounds on higher moments (and hence, for instance, variance) of T_ϵ are still questionable. It is at least trivial to show from our results that all higher moments are at least bounded. We believe the analysis performed here is a meaningful stepping stone to such analyses of higher moments. Directly related to this question, and of relevance to practical algorithms, the analysis performed here does not seem to immediately yield stopping criteria for a STORM algorithm. That is, although we can guarantee *expected* ϵ-stationarity in $\mathcal{O}(1/\epsilon^2)$ iterations, this does not suggest that any form of observed ϵ-criticality in a single realization of iterates yielded by a STORM algorithm should encourage the optimizer to stop the algorithm, as the measurements indicating ϵ-criticality are quite possibly the result of noise. Moreover, in the spirit of stochastic gradient methods, without better understanding the higher moments or tail distribution of T_ϵ, we have no strong theoretical basis to stop the algorithm within a predefined number $T \in \mathcal{O}(1/\epsilon^2)$ of iterations, since we can't quantify the probability that $T \geq T_\epsilon$ via this analysis alone. This issue is of great importance, and is of great theoretical and practical interest.

In Chapter 4, we considered the problem of generating α-probabilistically fully-linear models by analyzing a particular noise regime (σ-subgaussian noise) and limiting ourselves to regression models on various design stencils. Under different design stencils, with a particular interest in Monte Carlo sampling on a finite-difference coordinate stencil and random uniform sampling in a ball, we gave theoretical minimum sampling rates necessary to guarantee probabilistically fully linear models on a given ball. Most importantly, we showed that the minimum sampling rates for finite-difference coordinate stencils and random uniform sampling are roughly the same, only differing by a small numerical constant. As demonstrated in the computational experiment of this Chapter, however, we see a practical preference for random uniform sampling. In this Chapter, we also exhibited in this context a theoretical weakness of the standard strongly Λ-poised regression set definition in the DFO literature; in particular, when trying to yield meaningful bounds in this minimum

sampling context, the definition is simply too general and the obtained bounds are strictly worse.

In Chapter 5, we changed our focus to deterministic black-box nonsmooth functions, f. However, we assumed that the nonsmoothness was such that any query of a function value at a point would identify that point's membership to a smooth manifold of f. This led to yet another extension of the model-based trust-region framework for DFO problems that we called "manifold sampling". On each iteration, manifold sampling uses (possibly randomly) sampled manifold membership information to construct smooth "master models", which are treated like the usual smooth local models of trust-region methods, but with slight modifications in the acceptance test. We saw numerical evidence suggesting a preference for master models constructed using random queries (hence, yielding random models), and as such, this work fits within the scope of this thesis. Throughout this Chapter, we limited our attention to sums of absolute values of smooth (non-convex) black-box functions, because this is one instance in which manifold membership is immediate via sign patterns. In future work, it is of interest to find other interesting classes of problems that have not been previously studied for which manifold membership can be determined from the black-box. Another possible direction of future work is to extend the analysis to the case where evaluations of the component functions in the absolute sum are subject to stochastic noise, like in the assumptions of STORM, but we assume that we can build α-probabilistically fully-linear models of the individual component functions for an appropriate α.

Finally, in Chapter 6, we considered ℓ_1-regularized sigmoidal loss minimization problems from machine learning. We proposed HAILSTORM, an algorithm that can be seen as a variant of STORM that uses adaptive sampling methods to compute stochastic gradient models and function value estimates. HAILSTORM includes a manifold sampling step to handle the presence of the nonsmooth ℓ_1-regularization term, as opposed to the proximal points common in line-search methods. We did not prove any convergence results for HAILSTORM, but because the ℓ_1 term is deterministic given an iterate classifier w_k, the convergence of HAILSTORM should be straight-forward to demonstrate. This is a subject of future work.

Bibliography

[1] Z. Allen-Zhu and E. Hazan. Variance reduction for faster non-convex optimization. In *International Conference on Machine Learning*, 2016.

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

[3] F. Bach and E. Moulines. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems 24*, pages 451–459, 2011.

[4] A. S. Bandeira, K. Scheinberg, and L.N. Vicente. Convergence of trust-region methods based on probabilistic models. *SIAM Journal on Optimization*, 24(3):1238–1264, 2014.

[5] P.L. Bartlett, M.I. Jordan, and J.D. McAuliffe. Convexity, classification, and risk bounds. *Statistical Science*, 21(3):341–346, 2006.

[6] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[7] S. Ben-David, N. Eiron, and P.M. Long. On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, 66(3):496–514, 2003.

[8] S.C. Billups, P. Graf, and J. Larson. Derivative-free optimization of expensive functions with computational error using weighted regression. *SIAM Journal on Optimization*, 23(1):27–53, 2013.

[9] J. Blanchet, C. Cartis, M. Menickelly, and K. Scheinberg. Convergence rate analysis of a stochastic trust region method for nonconvex optimization. `https://arkiv.org/pdf/1609.07428`, 2016.

[10] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization Methods for Large-Scale Machine Learning. `https://arkiv.org/pdf/1606.04838`, 2016.

[11] J. V. Burke, A. S. Lewis, and M. L. Overton. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM Journal on Optimization*, 15(3):751–779, 2005.

[12] J.V. Burke. Descent methods for composite nondifferentiable optimization problems. *Mathematical Programming*, 33(3):260–279, 1985.

[13] R.H. Byrd, G.M. Chin, J. Nocedal, and Y. Wu. Sample size selection in optimization methods for machine learning. *Mathematical Programming*, 134(1):127–155, 2012.

[14] C.C. Chang and C.J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[15] K.H. Chang, M.K. Li, and H. Wan. Stochastic trust-region response-surface method (strong) - a new response-surface framework for simulation optimization. *INFORMS Journal on Computing*, 25(2):230–243, 2013.

[16] R. Chen. *Stochastic Derivative-Free Optimization of Noisy Functions*. PhD thesis, Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, USA, 2015.

[17] R. Chen, M. Menickelly, and K. Scheinberg. Stochastic optimization using a trust-region method and random models. *Mathematical Programming*. Advance online publication.

[18] F.H. Clarke. *Optimization and Nonsmooth Analysis*. John Wiley & Sons, 1983.

[19] A.R. Conn, K. Scheinberg, and L.N. Vicente. Geometry of sample sets in derivative-free optimization: polynomial regression and underdetermined interpolation. *IMA Journal of Numerical Analysis*, 28(4):721–748, 2008.

[20] A.R. Conn, K. Scheinberg, and L.N. Vicente. Global convergence of general derivative-free trust-region algorithms to first- and second-order critical points. *SIAM Journal on Optimization*, 20(1):387–415, April 2009.

[21] A.R. Conn, K. Scheinberg, and L.N. Vicente. *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009.

[22] F.E. Curtis and X. Que. An adaptive gradient sampling algorithm for non-smooth optimization. *Optimization Methods and Software*, 28(6):1302–1324, 2013.

[23] A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems 27*, pages 1646–1654, 2014.

[24] G. Deng and M.C. Ferris. Variable-number sample-path optimization. *Mathematical Programming*, 117:81–109, 2009.

[25] N. Ding and S.V.N. Vishwanathan. t-logistic regression. In *Advances in Neural Information Processing Systems 23*, pages 514–522, 2010.

[26] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[27] R. Durrett. Probability: theory and examples. cambridge series in statistical and probabilistic mathematics. *Cambridge University Press*, 105, 2010.

[28] F. Facchinei and J.S. Pang. *Finite-Dimensional Variational Inequalities and Complementarity Problems*. Springer-Verlag New York, Inc., New York, NY, 2003.

[29] R. Fletcher. A model algorithm for composite nondifferentiable optimization problems. In D. C. Sorensen and R. J.-B. Wets, editors, *Nondifferential and Variational Techniques in Optimization*, volume 17 of *Mathematical Programming Studies*, pages 67–76. Springer Berlin Heidelberg, 1982.

[30] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, New York, second edition, 1987.

[31] N. Cesa-Bianchi G. Cavallanti and C. Gentile. Learning noisy linear classifiers via adaptive and selective sampling. *Machine Learning*, 83(1):71–102, 2011.

[32] R. Garmanjani, D. Júdice, and L. N. Vicente. Trust-region methods without using derivatives: Worst case complexity and the non-smooth case. *SIAM Journal on Optimization*, 26(4):1987–2011, 2016.

[33] S. Ghadimi and G. Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[34] S. Ghadimi and G. Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming*, 156(1-2):59–99, 2016.

[35] S. Ghosh, P.W. Glynn, F. Hashemi, and R. Pasupathy. How much to sample in simulation-based stochastic recursions? *SIAM Journal on Optimization*, To appear.

[36] G.N. Grapiglia, J. Yuan, and Y.X. Yuan. A derivative-free trust-region algorithm for composite nonsmooth optimization. *Computational and Applied Mathematics*, pages 1–25, 2014.

[37] S. Gratton, C.W. Royer, L.N. Vicente, and Z. Zhang. Complexity and global rates of trust-region methods based on probabilistic models. Preprint 17-09, Dept. Mathematics, Univ. Coimbra, 2017.

[38] A. Griewank, A. Walther, S. Fiege, and T. Bosse. On Lipschitz optimization based on gray-box piecewise linearization. *Mathematical Programming*, pages 1–33, 2015.

[39] W. Hare and J. Nutini. A derivative-free approximate gradient sampling algorithm for finite minimax problems. *Computational Optimization and Applications*, 56(1):1–38, 2013.

[40] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26*, pages 315–323. 2013.

[41] A.B. Juditsky and B.T. Polyak. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, July 1992.

[42] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 22(3):462–466, 1952.

[43] K.C. Kiwiel. *Methods of Descent for Nondifferentiable Optimization*, volume 1133 of *Lecture Notes in Mathematics*. Springer Berlin Heidelberg, 1985.

[44] K.C. Kiwiel. A method for solving certian quadratic programming problems arising in nonsmooth optimization. *IMA Journal of Numerical Analysis*, (6):137–152, 1986.

[45] K.C. Kiwiel. Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization. *SIAM Journal on Optimization*, 18(2):379–388, 2007.

[46] K.C. Kiwiel. A nonderivative version of the gradient sampling algorithm for nonsmooth nonconvex optimization. *SIAM Journal on Optimization*, 20(4):1983–1994, 2010.

[47] G. Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133:365397, 2012.

[48] J. Larson and S. C. Billups. Stochastic derivative-free optimization using a trust region framework. *Computational Optimization and Applications*, 64(3):619645, 2016.

[49] J. Larson, M. Menickelly, and S.M. Wild. Manifold sampling for ℓ_1 nonconvex optimization. *SIAM Journal on Optimization*, 26(4):2540–2563, 2016.

[50] Y. LeCun, I. Kanter, and S.A. Solla. Second-order properties of error surfaces: learning time and generalization. In *Advances in Neural Information Processing Systems 3*, pages 913–924, 1991.

[51] M. Lichman. UCI machine learning repository, 2013. Data available at `http://archive.ics.uci.edu/ml`.

[52] J. Linderoth, A. Shapiro, and S. Wright. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*, 142(1):215–241, 2006.

[53] P. Long and R. Servedio. Random classification noise defeats all convex potential boosters. *Machine Learning*, 78(3):287–304, 2010.

[54] S. Monro and H. Robbins. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

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

[56] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[57] Y. Nesterov. Random gradient-free minimization of convex functions. CORE Discussion Papers 2011001, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2011.

[58] R. Pasupathy and S. Ghosh. Simulation optimization: A concise overview and implementation guide. In *TutORials in Operations Research*, chapter 7, pages 122–150. INFORMS, 2013.

[59] D. Popovic and A.R. Teel. Direct search methods for nonsmooth optimization. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, 2004.

[60] M.J.D. Powell. UOBYQA: unconstrained optimization by quadratic approximation. *Mathematical Programming*, 92(3):555–582, 2002.

[61] S.J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola. Stochastic variance reduction for nonconvex optimization. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 314–323, 2016.

[62] S.J. Reddi, S. Sra, B. Poczos, and A.J. Smola. Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. In *Advances in Neural Information Processing Systems 29*, pages 1145–1153. 2016.

[63] P. Richtarik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1,2):1–38, 2014.

[64] S.M. Robinson. Analysis of sample-path optimization. *Mathematics of Operations Research*, 21(3):513–528, 1996.

[65] R.T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898, 1976.

[66] S.M. Ross. *Stochastic processes*. Wiley series in probability and statistics: Probability and statistics. Wiley, 1996.

[67] A. Ruszczynski and A. Shapiro, editors. *Stochastic Programming*. Handbooks in Operations Research and Management Science, Volume 10. Elsevier, Amsterdam, 2003.

[68] S. Shashaani, F. S. Hashemi, and R. Pasupathy. Astro-df: A class of adaptive sampling trust-region algorithms for derivative-free simulation optimization. 2015. Under review.

[69] X. Shen, G.C. Tseng, X. Zhang, and W.H. Wong. On ψ-learning. *Journal of the American Statistical Association*, 101:500–509.

[70] J.C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37:332–341, 1992.

[71] J.C. Spall. Adaptive stochastic approximation by the simultaneous perturbation method. *Automatic Control, IEEE Transactions on*, 45(10):1839–1853, 2000.

[72] S.M. Wild. *Derivative-free optimization algorithms for computationally expensive functions*. Ph.D. thesis, Cornell University, 2009.

[73] S.M. Wild and C.A. Shoemaker. Global convergence of radial basis function trust-region algorithms for derivative-free optimization. *SIAM Review*, 55(2):349–371, 2013.

[74] Y.X. Yuan. Conditions for convergence of trust region algorithms for nonsmooth optimization. *Mathematical Programming*, (31):220–228, 1985.

Appendix A

Appendix

Algorithm 6: TR-SAA

(Initialization): Choose an initial point x_0 and an initial trust-region radius $\delta_0 \in (0, \delta_{\max})$ with $\delta_{\max} > 0$. Choose constants $\gamma > 1$, $\eta_1 \in (0,1)$, $\eta_2 \in (0, \infty)$, $p_{\max} = (n+1)(n+2)/2$, $p_{\min} \geq n+1$. Set $k \leftarrow 0$. Select some initial interpolation set $Y_0 \subset B(x_k, \delta_k)$ so that $|Y_0| \leq p_{\max}$ and $x_0 \in Y_0$. Compute an averaged function value estimate $\bar{f}(y) = \frac{1}{p_{min}} \sum_{i=1}^{p_{\min}} \tilde{f}(y, \omega_i)$ at each $y \in Y_0$.

while true do

  (Update sample rate): Set sample rate $p_k = \max\{p_{\min} + k, 1/\delta^2\}$.

  (Update interpolation value estimates): For each $y \in Y_k \cap Y_{k-1}$, compute $\bar{f}(y) = \frac{1}{p_k}[\sum_{i=p_{k-1}+1}^{p_k} \tilde{f}(y, \omega_i) + p_{k-1}\bar{f}(y)]$ and for each $y \in Y_k \setminus Y_{k-1}$, compute $\bar{f}(y) = \frac{1}{p_k} \sum_{i=1}^{p_k} \tilde{f}(y, \omega_i)$.

  (Model building): Build a quadratic model $m_k(x_k + s) = f_k + g_k^\top s + s^\top H_k s$ with $s = x - x_k$ that interpolates $\bar{f}(y)$ at the points of Y_k.

  (Step calculation): Compute $s_k = \arg \min_{s:\|s\| \leq \delta_k} m_k(s)$ (approximately) so that s_k satisfies (2.3).

  (Estimate calculation): Compute new estimate $f_k^s = \frac{1}{p_k} \sum_{i=1}^{p_k} \tilde{f}(x_k + s_k, \omega_i)$ of $f(x_k + s_k)$. Denote the current estimate of $f(x_k)$ by f_k^0.

  (Acceptance of the trial point): Compute $\rho_k = \dfrac{f_k^0 - f_k^s}{m_k(x_k) - m_k(x_k + s_k)}$.

  If $\rho_k \geq \eta_1$ and $\|g_k\| \geq \eta_2 \delta_k$, then $x_{k+1} \leftarrow x_k + s_k$; otherwise, $x_{k+1} \leftarrow x_k$.

  (Trust-region radius update): If $\rho_k \geq \eta_1$, $\delta_{k+1} \leftarrow \min\{\gamma\delta_k, \delta_{\max}\}$; otherwise $\delta_{k+1} \leftarrow \gamma^{-1}\delta_k$.

  (Interpolation set update): Augment Y_k with $x_k + s_k$. If $|Y_k| > p_{\max}$, remove the point of Y_k furthest from x_{k+1}.

  (Iterate): $k \leftarrow k + 1$.

end

Algorithm 7: STORM for unbiased noise

(Initialization): Choose an initial point x_0 and an initial trust-region radius $\delta_0 \in (0, \delta_{\max})$ with $\delta_{\max} > 0$. Choose constants $\gamma > 1$, $\eta_1 \in (0,1)$, $\eta_2 \in (0, \infty)$, $p_{\min}$; Set $k \leftarrow 0$. Select a regression set $Y_0 \subset B(x_k, \delta_k)$ satisfying $|Y_0| = p_{\min}$.

while true do

  (Update sample rate): Choose a sample rate $p_k = \max\{p_{\min} + k, 1/\delta^2\}$.

  (Regression set update): Uniformly sample a regression set $Y_k \subset B(x_k, \delta_k)$ satisfying $|Y_k| = p_k$.

  (Compute new regression value estimates): For each $y \in Y_k$, compute a single estimate $\tilde{f}(y, \omega)$.

  (Model building): Build a quadratic model $m_k(x_k + s) = f_k + g_k^\top s + s^\top H_k s$ with $s = x - x_k$ that regresses $\tilde{f}(y)$ at the points of Y_k.

  (Step calculation): Compute $s_k = \arg\min_{s:\|s\| \le \delta_k} m_k(s)$ (approximately) so that s_k satisfies (2.3).

  (Estimates calculation): Compute new estimates $f_k^0 = \sum_{i=1}^{p_k} \tilde{f}(x_k, \omega_i)$ and $f_k^s = \sum_{i=1}^{p_k} \tilde{f}(x_k + s_k, \omega_i)$ of $f(x_k)$ and $f(x_k + s_k)$.

  (Acceptance of the trial point): Compute $\rho_k = \dfrac{f_k^0 - f_k^s}{m_k(x_k) - m_k(x_k + s_k)}$.

  If $\rho_k \ge \eta_1$ and $\|g_k\| \ge \eta_2 \delta_k$, then $x_{k+1} \leftarrow x_k + s_k$; otherwise, $x_{k+1} \leftarrow x_k$.

  (Trust-region radius update): If $\rho_k \ge \eta_1$, $\delta_{k+1} \leftarrow \min\{\gamma \delta_k, \delta_{\max}\}$; otherwise $\delta_{k+1} \leftarrow \gamma^{-1} \delta_k$.

  (Iterate): $k \leftarrow k + 1$.

end

Algorithm 8: STORM for biased noise

(Initialization): Choose an initial point x_0 and an initial trust-region radius $\delta_0 \in (0, \delta_{\max})$ with $\delta_{\max} > 0$. Choose constants $\gamma > 1$, $\eta_1 \in (0, 1)$, $\eta_2 \in (0, \infty)$, $p_0 \geq n + 1$, $p_{\max} = (n + 1)(n + 2)/2$. Set $k \leftarrow 0$. Select an interpolation set $Y_0 \subset B(x_k, \delta_k)$ satisfying $|Y_0| = p_0$.

while true do

  (Compute new interpolation value estimates): For each $y \in Y_k$, compute a (new) estimate $\tilde{f}(y, \omega)$.

  (Model building): Build a quadratic model $m_k(x_k + s) = f_k + g_k^\top s + s^\top H_k s$ with $s = x - x_k$ that interpolates $\tilde{f}(y)$ at the points of Y_k.

  (Step calculation): Compute $s_k = \arg \min_{s:\|s\| \leq \delta_k} m_k(s)$ (approximately) so that s_k satisfies (2.3).

  (Estimates calculation): Compute new estimates $f_k^0 = \tilde{f}(x_k, \omega_i)$ and $f_k^s = \tilde{f}(x_k + s_k, \omega_i)$ of $f(x_k)$ and $f(x_k + s_k)$.

  (Acceptance of the trial point): Compute $\rho_k = \dfrac{f_k^0 - f_k^s}{m_k(x_k) - m_k(x_k + s_k)}$.

  If $\rho_k \geq \eta_1$ and $\|g_k\| \geq \eta_2 \delta_k$, then $x_{k+1} \leftarrow x_k + s_k$; otherwise, $x_{k+1} \leftarrow x_k$.

  (Trust-region radius update): If $\rho_k \geq \eta_1$, $\delta_{k+1} \leftarrow \min\{\gamma \delta_k, \delta_{\max}\}$; otherwise $\delta_{k+1} \leftarrow \gamma^{-1} \delta_k$.

  (Interpolation set update): Augment Y_k with $x_k + s_k$. If $|Y_k| > p_{\max}$, remove the point of Y_k furthest from the new trust region center x_{k+1}.

  (Iterate): $k \leftarrow k + 1$.

end

Algorithm 9: STORM for minimizing logistic loss

(Initialization): Choose an initial point $x_0 = (w_0, \beta_0)$ and an initial trust-region radius $\delta_0 \in (0, \delta_{\max})$ with $\delta_{\max} > 0$. Choose constants $\gamma > 1$, $\eta_1 \in (0, 1)$, $\eta_2 \in (0, \infty)$, p_0, $p_{\max}$; Set $k \leftarrow 0$.

while true do

  (Determine sample rate): Choose a sample rate p_k. In our implementation, we will use $p_k = \min\{p_{\max}, \max\{100 * k + p_0, \lceil 1/\delta_k^2 \rceil\}\}$.

  (Model building): Uniformly (without replacement) draw a sample $I_k \subset \{1, \ldots, N\}$. Compute a stochastic gradient $g_k = \nabla f_{I_k}(w, \beta)$ and stochastic Hessian $H_k = \nabla^2 f_{I_k}(w, \beta)$. Define a quadratic model $m_k(s) = g_k^\top s + \frac{1}{2} s^\top H_k s$. and stochastic Hessian

  (Step calculation): Compute $s_k = \arg \min_{s:\|s\| \le \delta_k} m_k(s)$ (approximately) so that s_k satisfies (2.3).

  (Estimates calculation): Draw new samples I_k^0, I_k^s and compute estimates $f_k^0 = f_{I_k^0}(x_k)$ and $f_k^s = f_{I_k^s}(x_k + s_k)$ of $f(x_k)$ and $f(x_k + s_k)$, respectively.

  (Acceptance of the trial point): Compute $\rho_k = \dfrac{f_k^0 - f_k^s}{m_k(x_k) - m_k(x_k + s_k)}$.

  If $\rho_k \ge \eta_1$ and $\|g_k\| \ge \eta_2 \delta_k$, then $x_{k+1} \leftarrow x_k + s_k$; otherwise, $x_{k+1} \leftarrow x_k$.

  (Trust-region radius update): If $\rho_k \ge \eta_1$, $\delta_{k+1} \leftarrow \min\{\gamma \delta_k, \delta_{\max}\}$; otherwise $\delta_{k+1} \leftarrow \gamma^{-1} \delta_k$.

  (Iterate): $k \leftarrow k + 1$.

end

Biography

Matt Menickelly was born in Portsmouth, New Hampshire on a fateful Thursday in 1988, and he still has far to go. Matt received undergraduate degrees in piano performance and mathematics from Miami University, with minors in Chinese language and actuarial science, in May 2010. He continued on to earn a Master of Science in Mathematics from Miami University in August 2012.

Matt began his studies at the Department of Industrial and Systems Engineering of Lehigh University in August 2012. While living in Bethlehem, Matt met his husband, Andrew (Andrés) Palomo, in September 2012 and married him in November 2015. While at Lehigh, Matt spent Summer 2014 and Summer 2015 at Argonne National Laboratory in the Mathematics and Computer Science (MCS) Division, and spent ten months beginning in May 2016 at the Business Analytics and Mathematical Sciences Department of IBM T.J. Watson Research Center. Matt will be joining the MCS Division at Argonne National Laboratory in June 2017 as a Postdoctoral Appointee.