A DERIVATIVE-FREE TRUST-REGION ALGORITHM FOR THE OPTIMIZATION OF FUNCTIONS SMOOTHED VIA GAUSSIAN CONVOLUTION USING ADAPTIVE MULTIPLE IMPORTANCE SAMPLING

ALVARO MAGGIAR*, ANDREAS WÄCHTER†, IRINA S. DOLINSKAYA‡, AND JEREMY STAUM§

Abstract. In this paper we consider the optimization of a functional F defined as the convolution of a function f with a Gaussian kernel. We propose this type of objective function for the optimization of the output of complex computational simulations, which often present some form of deterministic noise and need to be smoothed for the results to be meaningful.

We introduce a derivative-free algorithm that computes trial points from the minimization of a regression model of the noisy function f over a trust region. The regression model is constructed from function values at sample points that are chosen randomly around iterates and trial points of the algorithm. The weights given to the individual sample points in the regression problem are obtained according to an adaptive multiple importance sampling strategy. This has two advantages. First, it makes it possible to re-use all noisy function values collected over the course of the optimization. Second, the resulting regression model converges to the second-order Taylor approximation of the convolution functional F. We prove that, with probability one, each limit point of the iterates is a stationary point of F.

Computational experiments on a set of benchmark problems with noisy functions compare the proposed algorithm with the deterministic derivative-free trust-region method the proposed method is based on. It is demonstrated that the proposed algorithm performs similarly efficiently in the early stages of the optimization and is able to overcome convergence problems of the original method, which might get trapped in spurious local minima induced by the noise.

Key words. Derivative-free optimization; adaptive multiple importance sampling; trust region method; deterministic computational noise; Gaussian smoothing; Monte-Carlo sampling

AMS subject classifications. 90C15, 90C30, 90C56

1. Introduction.

In this paper we study an algorithm for solving the optimization problem

$$\min_{x \in C} F(x), \tag{1.1}$$

where $C \subseteq \mathbb{R}^n$ is a compact convex set, and the objective function is defined as the convolution of a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ with a Gaussian kernel ϕ_σ, i.e.,

$$F(x) := (f * \phi_\sigma)(x) = \int_{\mathbb{R}^n} f(y)\phi_\sigma(y; x)\mathrm{d}y, \tag{1.2}$$

where

$$\phi_\sigma(y; x) := \frac{1}{(\sqrt{2\pi})^n \sigma^n} \exp\left(-\frac{\|y - x\|^2}{2\sigma^2}\right) \tag{1.3}$$

*Amazon.com, Seattle, WA 98109, USA. This author was supported in part by the Office of Naval Research through the Autonomous Vehicle Dynamic Navigation System grant (N00014-11-1-0516). alvaro.maggiar@u.northwestern.edu.

†Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60208, USA. This author was supported in part by the National Science Foundation grants DMS-1216920 and DMS-1522747. andreas.waechter@northwestern.edu

‡National Science Foundation. This author was supported in part by the Office of Naval Research through the Autonomous Vehicle Dynamic Navigation System grant (N00014-11-1-0516). idolinsk@nsf.gov

§Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60208, USA. j-staum@northwestern.edu

is the probability density function (pdf) for the multivariate normal distribution with mean x and covariance matrix $\Sigma = \sigma^2 I$. The standard deviation $\sigma > 0$ is fixed. The convolution F can be interpreted as a smoothed version of the underlying function f, where σ determines the degree of smoothing. Note that F is infinitely many times differentiable, even when f is discontinuous.

In order for the integral (1.2) to be well-defined, the original function f need only satisfy mild assumptions since the Gaussian kernel belongs to the class of rapidly decaying functions. More precisely, we assume that f has sub-exponential growth, i.e., there exist constants $K_f > 0$ and $a_f > 0$ so that for all $y \in \mathbb{R}^n$ we have

$$|f(y)| \leq K_f \exp(a_f \|y\|). \tag{1.4}$$

For instance, any function that is the product of polynomials with elements of $L^p(\mathbb{R}^n)$ $(p \geq 1)$ is valid.

We assume here that the integral (1.2) cannot be computed directly. Instead, we approximate F based on evaluations of f at some sample points $\{y_j\}$, exploiting the equivalent characterization of the objective function as an expectation,

$$F(x) = \mathbb{E}_{x,\sigma}[f(Y)], \tag{1.5}$$

where the subscript x, σ denotes that in this expectation, the pdf of the random variable Y is given by $\phi_\sigma(\cdot, x)$. Our method makes use of quadratic models obtained by performing a weighted regression to approximate the smoothed function F close to a current iterate x_k. The choice of the weights in the regression is based on a Monte-Carlo approximation of the expectation (1.5). A straight-forward Monte-Carlo approximation, however, would require the generation of a new set of sample points from the normal distribution $\phi_\sigma(\cdot, x_k)$ at each point x_k.

The key contribution of this paper lies in the use of *adaptive multiple importance sampling* [8, 41, 46] which permits the re-use of all previously computed function values $f(y_j)$ despite the fact that the distribution of the random variable Y changes with x_k.

The regression procedure can become expensive by itself as the dimension of the problem increases, and our algorithm is thus designed for problems of relatively small dimension.

1.1. Motivation: Deterministic Computational Noise. The motivation of this work stems from the optimization of the output $f(x)$ of complex computational simulations, which are often burdened by the presence of deterministic computational noise. This noise can be many orders of magnitude larger than the truncation error due to numerical precision. It is caused, for example, by tolerance thresholds that lead to a varying number of inner iterations during the solution of linear systems of equations, or by the adaptation of the grid discretization in partial differential equation solvers. Another source can be the use of piecewise constant approximations of continuous functions to accelerate the computation, for example in a simulator of vessel roll of contained ships [50] and in transistor-level circuit tuning [12, 47]. All these cases have in common that the noise has a deterministic nature in the sense that when repeatedly evaluated at a given point x, the function value $f(x)$ will remain the same, unlike in the case of random noise (e.g., in discrete event simulation) where repeated sampling can be leveraged to increase the precision of the function estimate.

Typically, the deterministic computational noise manifests itself in the form of discontinuities in the objective function f. For example, the output of an iterative

linear solver depends smoothly on the problem data (up to the usual finite-precision round-off) except at points where the number of inner iterations changes. This can become a significant obstacle for common optimization techniques which typically rely on smoothness assumptions such as continuity or differentiability.

Strictly speaking, when the simulation output f is not lower semicontinuous, the task of minimizing f can even be mathematically ill-defined because a minimizer might not exist. Also, the piecewise smooth nature of the objective function gives rise to many artificial local minima that are merely an artifact of the complex simulation and do not reflect any natural phenomena. The simulation output is only precise up to a certain level, and the variations caused by the different execution pathways through the computation are deemed negligible by the application modeler who sets appropriate tolerances in the simulator. It therefore makes sense to approximate the raw simulation output f by a smooth function that still captures the relevant behavior of the simulated system. We propose to do this with the convolution F (1.2). In this way, we also obtain a mathematically meaningful optimization problem with well-defined solutions.

The effect of deterministic noise in convex optimization has been investigated in [35]; the estimation of numerical noise and its consequences on finite-difference gradients was studied in [33, 34], while its impact in flow optimization problems, for example, has been considered in [7, 48].

A number of methods have been proposed for the optimization of noisy functions. These include in particular Implicit Filtering [25, 26], Simplex [36] and Pattern Search methods [23], but the convergence results of these algorithms require that the noise of the function can be reduced to zero about a local minimum. We do not, however, assume that the noise satisfies any structure or specific property.

1.2. Smoothing. As convolution with a Gaussian kernel (1.2), $F(x)$ can be interpreted as a local weighted average of function values $f(z)$ at points z close to x. The parameter σ determines the degree of the smoothing, since it defines the standard deviation of the normal pdf ϕ_σ and therefore the size of the neighborhood around x that is relevant for the computation of $F(x)$. As σ goes to 0, the Gaussian kernel tends to the Dirac delta function in the weak sense. Consequently, $\lim_{\sigma\to 0} F(x) = f(x)$ whenever f is continuous at x, and we recover the original function. On the other hand, when σ is large, local features of f are smoothed out. The Ph.D. thesis [28] demonstrates this effect for the output of a vessel roll simulator [1, 39, 40, 50].

In general, the difference between the underlying function f and its smooth approximation F depends on a number of factors. For example, assuming that $f(x) = \bar{f}(x)$, where $\bar{f}$ is a function that has Lipschitz-continuous gradient with constant L, it is easily proved that the following inequalities hold:

$$\bar{f}(x) - \frac{nL\sigma^2}{2} \le F(x) \le \bar{f}(x) + \frac{nL\sigma^2}{2},$$
$$\left\|\nabla\bar{f}(x) - \nabla F(x)\right\| \le \sqrt{n}L\sigma.$$

If we assume that the underlying function has sinussoidal noise with angular frequencies ω, i.e., $f(x) = \bar{f}(x) + c\,\sin(\omega^T y)$ for some constant c, then the smoothed noise $\int \sin(\omega^T y)\phi_\sigma(y;x)\mathrm{d}y$ is related to the Fourier transform of the multivariate Gaussian, which is proportional to $e^{-\|\omega\|^2/2\sigma^2}$. As a consequence, the bounds above are approached exponentially quickly as the frequency of the noise increases.

For the purpose of this paper, we assume that the parameter σ is fixed throughout and exogenous. That is, the user provides a value of σ that is deemed appropriate

for the current application. Computational noise often presents itself in the form of jumps in the function values that occur with some regularity and are of similar size. In this case, we propose as a rule of thumb that σ should be slightly larger than the apparent periodicity of the noise. This periodicity can be estimated with few function evaluations using the method described in [33]. Our algorithm can be extended by letting the target smoothing parameter σ vary during the course of the algorithm to allow for an adaptive choice.

1.3. Derivative-Free Optimization. In this paper we assume that the computer simulation for a given input x computes only the output $f(x)$ and does not compute derivatives of f at x. A number of derivative-free optimization (DFO) methods have been designed for this setting (see [17] for a comprehensive overview or [19] for a more recent and shorter one). Our method builds on the model-based DFO framework [14, 16] in which a local quadratic model is computed at each iterate x_k and used for the computation of the next iterate. This framework has proven efficient and robust in practice, even in the presence of moderate deterministic noise [32]. Our method strives to display a similar behavior in the early optimization iterations when the noise of the function is insignificant with respect to the size of the steps taken.

In standard model-based DFO methods, the quadratic model is a local approximation of f around the current iterate x_k. It is typically obtained by interpolation and matches the model with the function values $\{f(y_j)\}$ at sample points $\{y_j\}$. In the presence of noise, regression is more appealing than interpolation because it has the tendency to smooth out the noise. In that case, however, the weights given to the individual sample points must be chosen with care: (i) points far from the current iterate x_k should receive less weight than points close by; and (ii) points that appear in a cluster should individually be given less weight so that the cluster does not dominate and bias the regression.

By defining the objective function as expectation (1.5) we are able to obtain the regression weights in a natural and consistent manner that permits global convergence guarantees. We do so by applying adaptive multiple importance sampling [8, 41, 46], a technique originally developed for variance reduction in Monte-Carlo simulation. One key result for our analysis shows the convergence of the regression model to the second-order Taylor expansion of the function F at the current iterate; see Lemma 2.1. It is important to note that this result crucially depends on the particular choice of the smoothing function ϕ_σ as the pdf of the normal distribution. Our analysis does not hold for other kernels.

The expected-value formulation (1.5) highlights the fact that the proposed algorithm belongs to the class of stochastic optimization algorithms. From this point of view, the proposed method can be regarded as a stochastic DFO method specialized to the solution of the specific stochastic function (1.5).

For general stochastic objective functions, several variations of the deterministic DFO method have been proposed recently. In [4], Bendeira, Scheinberg, and Vicente generalize the trust-region framework to enable the use of stochastic local models m_k that satisfy a probabilistic version of the "κ-fully linear/quadratic model" condition. In essence, this condition ensures that the model approximates, with a given probability, the true function increasingly more accurately as the trust region radius converges to zero. While in [4] it is assumed that the function itself can be computed exactly, the more recent work by Chen, Menickelly, and Scheinberg [10] extends this framework so that only estimates of the function values are required, as long as these estimates, with a given probability, are increasingly more accurate as the trust region shrinks to

zero. This framework is further analyzed in [6]. Also based on [4], Larson and Billups [27] developed a related method for a similar setting. The stochastic trust-region DFO method proposed by Shashaani et al. [45] has explicit rules to adjust the number of replications for the function estimates based on observed variances.

It is important to highlight that the algorithm proposed in this paper is tailored to the minimization of the specific objective function (1.2) which is defined as a convolution. It is therefore not in direct competition with the aforementioned methods. On the other hand, while those methods could be applied to the minimization of the stochastic function (1.5), they do not take advantage of the fact that samples drawn for the estimation of $F(x)$ at a particular point x can be reused for the approximation at a neighboring point.

Specifically targeting objective functions with computational error, Billups, Larson, and Graf [5] discuss a variant of the deterministic DFO framework in which the trust region model m_k is obtained by regression with the goal of smoothing the noise. The weighing parameters for the regression problem are determined by a heuristic. In contrast, we propose a theoretically founded choice of the regression weights guided by an unambiguous definition of a smooth objective (1.2).

Finally, we point out that Deng and Ferris [20] adapted Powell's derivative-free UOBYQA algorithm [43] to functions with stochastic noise. This method relies on the use of common random number generators and is not suitable for the optimization of functions with deterministic noise.

1.4. Notation and Structure of the Paper. With $\|v\|$ for a vector v we denote a fixed vector norm, and with $\|A\|$ for a matrix A the corresponding induced matrix norm. We write $v^{(i)}$ for the ith component of a vector v. For a given square matrix A, $\operatorname{cond}(A)$ denotes the condition number of A and $\operatorname{Tr}(A)$ denotes the trace of A. For a finite set $\mathcal{Y}$, we write $|\mathcal{Y}|$ for the number of elements in the set. Finally, we write $Y \sim \mathcal{N}(y, \Sigma)$ to indicate that Y is a multi-variate normal random variable with mean y and covariance matrix Σ.

The paper is structured as follows. In Section 2, we introduce the components of the proposed algorithm which is stated formally in Section 2.5. The convergence properties of the method are analyzed in Section 3. We present some numerical results in Section 4 and conclude the paper in Section 5.

2. Description of the Algorithm.

2.1. Trust-Region Framework. The algorithm proposed in this paper is based on the trust-region framework [13, 38]. At iteration k, given the current iterate x_k, a quadratic model

$$m_k(x_k + p) := b_k + g_k^T p + \frac{1}{2} p^T H_k p \approx F(x_k + p) \tag{2.1}$$

is built that approximates the objective function $F(x)$ around x_k. In the case where derivative information is available, the model gradient g_k is usually set equal to the objective function gradient $\nabla F(x_k)$ at x_k and the model Hessian H_k to the Hessian of the objective function $\nabla^2 F(x_k)$, or an approximation thereof if second-order information is not available. In the absence of derivative information, the model is obtained in a different manner, most notably using interpolation or regression techniques [17]. Our approach follows that line of work, and the details of our approach to the model construction are described in Section 2.3.

Once the model (2.1) has been constructed, it is minimized within a trust-region centered at x_k and of radius Δ_k,

$$\min_{p\in\mathbb{R}^n} \; m_k(x_k+p) \tag{2.2a}$$

$$\text{s.t. } \|p\| \leq \Delta_k, \tag{2.2b}$$

$$x_k + p \in C, \tag{2.2c}$$

where $\|\cdot\|$ denotes the Euclidean norm. The solution p_k of this subproblem provides the trial point $s_k = x_k + p_k$ for a potential new iterate. Note that (2.2c) implies that s_k is feasible for (1.1).

To ensure convergence, trust-region methods typically require only an approximate solution p_k to (2.2) so long as it yields sufficient decrease in the model. Here, we require that p_k satisfies the Generalized Cauchy decrease condition [13]

$$m_k(x_k) - m_k(x_k+p_k) \geq \frac{1}{2}\chi_k \min\left\{\frac{\chi_k}{\|H_k\|}, \Delta_k\right\} =: \xi_k, \tag{2.3}$$

where $\chi_k = \tilde{\chi}(g_k, x_k)$ with

$$\tilde{\chi}(g,x) = \min\{g^T d : x + d \in C, \|d\| \leq 1\}, \tag{2.4}$$

and we let $\chi_k/\|H_k\|$ be $+\infty$ if $\|H_k\| = 0$. As the value function of a convex optimization problem, $\tilde{\chi}$ is continuous in g and x [13, Theorem 3.2.8], and therefore so is $\chi(x) = \tilde{\chi}(\nabla F(x), x)$. It has the property that $\chi(x_*) = 0$ if and only if x_* is a first-order stationary point for (1.1) [13, Theorem 12.1.6]. The proposed algorithm seeks to find points satisfying this condition.

Condition (2.3) is satisfied by the Generalized Cauchy Point as defined in [13, Section 12.2.1]. This reference also describes an algorithm for the computation of this point. In particular, the exact solution of (2.2) satisfies (2.3).

Once an approximate solution p_k of the subproblem (2.2) has been found, the quality of the trial point is evaluated, using the ratio

$$\rho_k := \frac{F(x_k) - F(x_k+p_k)}{m_k(x_k) - m_k(x_k+p_k)} \tag{2.5}$$

of the actual decrease in the function over the predicted decrease in the model. A value of ρ_k close to one is an indication of a good agreement between the model and the objective function and of good reduction of the objective function. On the other hand, a value close to zero or negative indicates that the model m_k is a poor approximation of the function over the trust-region. The value of ρ_k dictates whether the tentative new iterate s_k is accepted, and how the trust-region radius Δ_k should be updated.

The challenge in our context is that only estimates of the objective function $F(x)$ defined in (1.2) are available so that ρ_k is known only approximately. Section 2.4 discusses in detail how this approximation is computed and improved when necessary.

2.2. Monte-Carlo approximation and adaptive multiple importance sampling. The novelty of our method lies in the use of samples from previous iterations, thereby reducing the number of evaluations, or samples, of the black-box function f. This is achieved using Monte-Carlo simulation and importance sampling [22].

Our approach relies on the computation of estimates for integrals of the type

$$G(x) = \int g(y)\phi_\sigma(y;x)\mathrm{d}y$$

for a given function $g : \mathbb{R}^n \longrightarrow \mathbb{R}$. These will play a crucial role in the model construction (Section 2.3) and acceptance test (Section 2.4). If we regard $\phi_\sigma(\,\cdot\,;x)$ as a probability density function, then $G(x)$ can be interpreted as

$$G(x) = \mathbb{E}_{x,\sigma}\left[g(Y)\right],$$

the expectation of a function g of a normal random variable $Y \sim \mathcal{N}(x, \sigma^2 I)$. A straight-forward Monte-Carlo approximation of $G(x)$ is obtained by sampling a number of points $\{y_i\}$ from the distribution $\phi_\sigma(\,\cdot\,;x)$, and computing the sample average approximation (SAA) of g evaluated at those points,

$$G(x) \approx \frac{1}{N}\sum_{i=1}^{N} g(y_i).$$

Importance sampling, originally introduced for variance reduction of Monte-Carlo estimates, makes it possible to use points that have been sampled from a distribution other than the target distribution ϕ_σ in order to estimate $G(x)$. It is derived by rewriting $G(x)$ as an expectation with respect to a different random variable $\tilde{Y}$ that has pdf ψ:

$$G(x) = \int g(y)\phi_\sigma(y;x)\mathrm{d}y = \int g(\tilde{y})\frac{\phi_\sigma(\tilde{y};x)}{\psi(\tilde{y})}\psi(\tilde{y})\mathrm{d}\tilde{y} = \mathbb{E}\left[\frac{\phi_\sigma(\tilde{Y};x)}{\psi(\tilde{Y})}g(\tilde{Y})\right].$$

Given a finite set of points $\mathcal{Y} = \{y_1, \ldots, y_N\}$ sampled from ψ, we may then approximate $G(x)$ by the weighted average

$$\tilde{G}^{\mathcal{Y}}(x) = \frac{1}{N}\sum_{i=1}^{N} w_\sigma(y_i;x)g(y_i) \tag{2.6}$$

with weights $w_\sigma(y_i;x) = \frac{\phi_\sigma(y_i;x)}{\psi(y_i)}$. In standard importance sampling, the sampling distribution ψ is chosen in advance, and all the points in $\mathcal{Y}$ are sampled from the same distribution.

We use a variant of this technique, called *adaptive multiple importance sampling* [8]. Here, the sample points y_i are drawn from different (multiple) distributions ψ_i, and those distributions can be adaptive in the sense that they are not predetermined but chosen depending on earlier (random) samples. In this case, the weights in (2.6) become $w_\sigma(y_i;x) = \frac{\phi_\sigma(y_i;x)}{\psi_i(y_i)}$. Because the distributions are now themselves random variables, the analysis becomes considerably more complicated, as underlined by Conuet et al. [18]. Within the context of Monte-Carlo estimation, adaptive multiple importance sampling has been used to learn good parameters for the sampling distribution, with the goal of finding a distribution that minimizes the variance of the estimator. In the literature on adaptive multiple importance sampling stemming from Arouna [2], it is usually assumed that old samples are discarded once a new distribution has been selected.

In our algorithm, a sample point y_i is drawn from the normal distribution $\psi_i(\,\cdot\,) = \phi_\sigma(\,\cdot\,;t_i)$, where the mean t_i is either an iterate x_k or a trial point s_k. Let $\mathcal{Y} =$

$\{(t_1, y_1), \ldots, (t_{|\mathcal{Y}|}, y_{|\mathcal{Y}|})\}$ be a set of sample points. For bookkeeping purposes, we store the samples y_i together with the means t_i of the distributions they were sampled from. Applying adaptive multiple importance sampling then results in the estimate

$$\tilde{G}^{\mathcal{Y}}(x) = \frac{1}{|\mathcal{Y}|} \sum_{(t_i,y_i)\in\mathcal{Y}} w_\sigma(t_i, y_i; x) g(y_i) \quad \text{with} \quad w_\sigma(t_i, y_i; x) = \frac{\phi_\sigma(y_i; x)}{\phi_\sigma(y_i; t_i)} \tag{2.7}$$

of the integral $G(x)$. The weights $w_\sigma(t_i, y_i; x)$ have the form of likelihood ratios [41, 42]. It can be shown that the estimates are consistent, i.e., for a fixed x, $\tilde{G}^{\mathcal{Y}}(x)$ converges almost surely to $G(x)$ as the sample set $\mathcal{Y}$ is suitably augmented. A detailed discussion of these convergence properties is given in Section 3.1.

Note that the weights $w_\sigma(t_i, y_i; x)$ might not add up to one. To yield more reliable estimates of the integral $G(x)$ when needed, we also use self-normalized weights [8, 42],

$$\bar{G}^{\mathcal{Y}}(x) = \sum_{(t_i,y_i)\in\mathcal{Y}} \bar{w}_\sigma(t_i, y_i; x) g(y_i) \quad \text{with} \quad \bar{w}_\sigma(t_i, y_i; x) := \frac{w_\sigma(t_i, y_i; x)}{\sum_{(t_j,y_j)\in\mathcal{Y}} w_\sigma(t_j, y_j; x)}, \tag{2.8}$$

so that $\bar{G}^{\mathcal{Y}}(x)$ becomes a weighted average.

2.3. Model Construction. Our algorithm requires the quadratic model (2.1) to compute the trial step. In traditional trust-region methods, b_k, g_k and H_k are approximations of the value of the smoothed function F, its gradient ∇F and its Hessian $\nabla^2 F$ at x_k, respectively, so that $m_k(x_k + p)$ represents an approximate Taylor expansion of F about x_k. A consequence of the definition of F as a convolution with a Gaussian kernel is that F inherits the smoothness properties of the kernel function even though f itself might not be continuous. F is therefore infinitely many times differentiable. Furthermore, we can write simple expressions for the derivatives of F. Indeed, using Leibniz' rule it is not difficult to prove the following relations:

$$\nabla F(x) = \int \frac{(y-x)}{\sigma^2} \phi_\sigma(y; x) f(y) \mathrm{d}y,$$
$$\nabla^2 F(x) = \int \left[\frac{(y-x)(y-x)^T}{\sigma^2} - I \right] \frac{\phi_\sigma(y; x)}{\sigma^2} f(y) \mathrm{d}y.$$

The expressions may be stated equivalently in the following way:

$$F(x) = \mathbb{E}\left[f(x + \sigma Z) \right], \tag{2.9a}$$
$$\nabla F(x) = \tfrac{1}{\sigma} \mathbb{E}\left[f(x + \sigma Z) Z \right], \tag{2.9b}$$
$$\nabla^2 F(x) = \tfrac{1}{\sigma^2} \mathbb{E}\left[f(x + \sigma Z) \left(Z Z^T - I \right) \right], \tag{2.9c}$$

where Z is the standard multivariate normal random variable (with variance 1). We could obtain estimates of these quantities directly using the Monte-Carlo approximation and importance sampling techniques described in the previous section. An alternative way to obtain the model m_k is to perform some form of regression. In many model-based derivative-free optimization methods, the model m_k is computed by interpolating or regressing points within a region that is usually tied to the trust-region used in the subproblem (2.2). Such methods include DFO [14], Wedge [29] and UOBYQA [43].

Here we formulate a local regression problem [11] from the solution of which we can derive the values of $g = \nabla F(x)$ and $H = \nabla^2 F(x)$ in the quadratic model

$$m_\zeta(y; x) = b + g^T (y - x) + \frac{1}{2} (y - x)^T H (y - x). \tag{2.10}$$

In abuse of notation, the parameter vector $\zeta := (b, g, H)$ consists of the scalar b, the elements of the vector g, and the n^2 matrix elements of H. Our approach minimizes a weighted error between the function f and the quadratic model m_ζ. In consistency with (1.2), we use the Gaussian kernel ϕ_σ as the weight function. This leads to the local quadratic regression problem

$$\min_{\zeta=(b,g,H)} \int (m_\zeta(y;x) - f(y))^2 \, \phi_\sigma(y;x) \mathrm{d}y. \tag{2.11}$$

It turns out that solving problem (2.11) indeed yields the desired quantities, as the following lemma states.

LEMMA 2.1. *The optimal solution of the regression problem* (2.11) *is given by* $b = F(x) - \frac{\sigma^2}{2} Tr(H)$, $g = \nabla F(x)$, *and* $H = \nabla^2 F(x)$.

Proof. First note that the problem (2.11) can be equivalently stated as

$$\min_{\zeta} \mathbb{E}\left[(f(x+\sigma Z) - m_\zeta(x+\sigma Z; x))^2\right],$$

where we recall that $Z \sim \mathcal{N}(0, I)$. In order to determine the optimal solution to problem (2.11), using Leibniz' rule one can derive the first-order optimality conditions

$$\begin{aligned}
\mathbb{E}\left[f(x+\sigma Z) - m_\zeta(x+\sigma Z; x)\right] &= 0, && \text{(w.r.t. } b\text{)} \\
\mathbb{E}\left[(f(x+\sigma Z) - m_\zeta(x+\sigma Z; x))Z\right] &= 0, && \text{(w.r.t. } g\text{)} \\
\mathbb{E}\left[(f(x+\sigma Z) - m_\zeta(x+\sigma Z; x))ZZ^T\right] &= 0. && \text{(w.r.t. } H\text{)}
\end{aligned}$$

Using the following properties of Gaussian random vectors: $\mathbb{E}[Z] = 0$, $\mathbb{E}[Z^T H Z] = \mathrm{Tr}(H)$ and $\mathbb{E}[ZZ^T] = I$, we may transform these conditions. The first condition can be rewritten as

$$\mathbb{E}\left[f(x+\sigma Z)\right] = \mathbb{E}[m_\zeta(x+\sigma Z; x)] = b + \sigma \mathbb{E}\left[g^T Z\right] + \frac{\sigma^2}{2}\mathbb{E}\left[Z^T H Z\right] = b + \frac{\sigma^2}{2}\mathrm{Tr}(H). \tag{2.12}$$

Then (2.9a) implies $b = F(x) - \frac{\sigma^2}{2}\mathrm{Tr}(H)$. The second condition is

$$\begin{aligned}
\mathbb{E}\left[f(x+\sigma Z)Z\right] &= \mathbb{E}[m_\zeta(x+\sigma Z; x)Z] = \mathbb{E}\left[bZ\right] + \sigma \mathbb{E}\left[(g^T Z)Z\right] + \frac{\sigma^2}{2}\mathbb{E}\left[(Z^T H Z)Z\right] \\
&= \sigma \mathbb{E}\left[(g^T Z)Z\right] = \sigma g,
\end{aligned}$$

which, together with (2.9b), yields $g = \nabla F(x)$. The third condition reads

$$\mathbb{E}\left[f(x+\sigma Z)ZZ^T\right] = \mathbb{E}[m_\zeta(x+\sigma Z; x)ZZ^T] = bI + \frac{\sigma^2}{2}\mathbb{E}[(Z^T H Z)ZZ^T]. \tag{2.13}$$

Using elementary properties of the Gaussian random variables similarly as above, the second term on the right is given by $\mathbb{E}[(Z^T H Z)ZZ^T] = 2H + \mathrm{Tr}(H)I$. Substituting this and (2.12) into (2.13), we obtain

$$\begin{aligned}
\mathbb{E}\left[f(x+\sigma Z)ZZ^T\right] &= \mathbb{E}[f(x+\sigma Z)]I - \frac{\sigma^2}{2}\mathrm{Tr}(H)I + \frac{\sigma^2}{2}\mathrm{Tr}(H)I + \sigma^2 H \\
&= \mathbb{E}[f(x+\sigma Z)]I + \sigma^2 H.
\end{aligned}$$

Together with (2.9c), this proves $H = \nabla^2 F(x)$. ∎

The proof above relies on specific properties of standard normal random variables. This is the reason behind the choice of the Gaussian kernel in our algorithm.

This result implies that we are able to recover the values of $F(x)$, $\nabla F(x)$ and $\nabla^2 F(x)$ from (2.11). However, this problem cannot be solved directly because the integral cannot be computed. Nevertheless, applying the adaptive multiple importance sampling technique described in the previous section with a sample set $\mathcal{Y}$ in iteration k suggests the following sample average approximation of the regression problem:

$$\min_{\zeta} \frac{1}{|\mathcal{Y}|} \sum_{(t_i,y_i)\in\mathcal{Y}} w_\sigma(t_i, y_i; x_k)\big(f(y_i) - m_\zeta(y_i; x_k)\big)^2 \tag{2.14}$$

where the weights $w_\sigma(t_i, y_i; x_k)$ are given in (2.7). Its optimal solution $\zeta_k = (b_k, g_k, H_k)$ is used in our algorithm to define the quadratic model (2.1), with $m_k(x_k + p) = m_{\zeta_k}(x_k + p; x_k)$. We note that this is different from the regression approach in other model-based DFO methods, where each term in the sum has equal weight [14] or in which the weights are obtained by some heuristic [5].

We remark that we chose to define the objective function in (2.14) in terms of the non-normalized weights (2.7). This might appear as an inconsistency to the definition of the approximation of $F(x)$ in (2.20) below, which uses the normalized weights (2.8). However, since scaling the objective function in (2.14) does not change the optimal solution, the normalized and non-normalized formulations are equivalent. Using the non-normalized formulation (2.14) simplifies the analysis.

The objective function in (2.14) is a convex quadratic function, and its minimizer can be computed by solving a linear system of equations. For numerical reasons it is often preferable to scale this linear system [13, p.49], with $\hat{\zeta} = (\hat{b}, \hat{g}, \hat{H})$, where $\hat{b} = b$, $\hat{g} = \frac{1}{\sigma} g$, and $\hat{H} = \frac{1}{\sigma^2} H$.

Let $W_\sigma^{\mathcal{Y}}$ be the $|\mathcal{Y}|$-dimensional diagonal matrix with the weights $w_\sigma(t_i, y_i; x_k)/|\mathcal{Y}|$ on the diagonal. The regression problem (2.14) can be expressed in matrix form as

$$\min_{\hat{\zeta}} \left\| (W_\sigma^{\mathcal{Y}})^{\frac{1}{2}} \left(M_\sigma^{\mathcal{Y}}(x_k)\hat{\zeta} - f(\mathcal{Y}) \right) \right\|_2^2, \tag{2.15}$$

where $M_\sigma^{\mathcal{Y}}(x_k)$ is a matrix whose ith row corresponds to $(t_i, y_i) \in \mathcal{Y}$ and is given by

$$[M_\sigma^{\mathcal{Y}}(x_k)]_i = \left(1, z_{i1}, \ldots, z_{in}, \frac{z_{i1}^2}{2}, z_{i1}z_{i2}, \ldots, z_{i1}z_{in}, z_{i2}z_{i1}, \frac{z_{i2}^2}{2}, \ldots, z_{i(n-1)}z_{in}, \frac{z_{in}^2}{2}\right) \tag{2.16}$$

with $z_{ij} = \frac{y_i^{(j)} - x_k^{(j)}}{\sigma}$, and where $f(\mathcal{Y}) := \big(f(y_1), \ldots, f(y_{|\mathcal{Y}|})\big)^T$. Defining

$$K_\sigma^{\mathcal{Y}}(x_k) := M_\sigma^{\mathcal{Y}}(x_k)^T W_\sigma^{\mathcal{Y}} M_\sigma^{\mathcal{Y}}(x_k) \quad \text{and} \quad r_\sigma^{\mathcal{Y}}(x_k) := M_\sigma^{\mathcal{Y}}(x_k)^T W_\sigma^{\mathcal{Y}} f(\mathcal{Y}), \tag{2.17}$$

the solution of (2.15) is obtained by solving the linear system

$$K_\sigma^{\mathcal{Y}}(x_k)\hat{\zeta}_k = r_\sigma^{\mathcal{Y}}(x_k). \tag{2.18}$$

Writing $\hat{\zeta}_k = (\hat{b}_k, \hat{g}_k, \hat{H}_k)$, we recover the solution ζ_k of (2.14) as $\zeta_k = (\hat{b}_k, \sigma\hat{g}_k, \sigma^2\hat{H}_k)$.

To guarantee that the model is well-defined, the algorithm must ensure that the matrix $K_\sigma^{\mathcal{Y}}(x_k)$ is sufficiently well-conditioned. Model-based DFO methods such as those discussed in [5, 17] guarantee this by making sure that the interpolation or regression points are well-poised. This requires checking and sometimes improving

the geometry of the interpolation points, and derivative-free optimization algorithms usually incorporate an additional step to perform these tasks.

In our method, the matrix $K_\sigma^{\mathcal{Y}}(x_k)$ might be close to singular not only because of the geometry of the points, but also because the weights $w_\sigma(t_i, y_i; x_k)$ are very small when $\|y_i - x_k\| \gg \sigma$. For example, suppose that an iteration takes a large step, so that x_k is far from the previous points, compared to σ, and that a single point is sampled from $\phi_\sigma(\,\cdot\,; x_k)$. This one new point then has a large weight while all other points have essentially zero weights. The situation is similar to trying to fit a quadratic model using a single point. We address this issue by sampling additional points according to $\phi_\sigma(\,\cdot\,; x_k)$, until the condition number of $K_\sigma^{\mathcal{Y}}(x_k)$ is below a fixed threshold $\kappa_{\max}$. Lemma 3.4 below shows this can be achieved by adding sufficiently many sample points, as long as $\kappa_{\max} > \kappa_n$, where

$$\kappa_n = \frac{n + 6 + \sqrt{n^2 + 12n + 4}}{n + 6 - \sqrt{n^2 + 12n + 4}}. \tag{2.19}$$

Reducing the condition number by adding more sample points might require a lot of function evaluations, especially in higher dimensions. Section 4.1.2 describes an alternative that temporarily increases the standard deviation σ of the target distribution for the weights in the regression problem (2.14).

2.4. Step Acceptance Test. A basic trust-region algorithm updates the trust-region radius and the iterate according to

$$x_{k+1} = \begin{cases} s_k = x_k + p_k & \text{if } \rho_k \geq \eta \\ x_k & \text{if } \rho_k < \eta \end{cases},$$

$$\Delta_{k+1} = \begin{cases} \min\{\gamma_{\text{inc}}\Delta_k, \Delta_{\max}\} & \text{if } \rho_k \geq \eta \\ \gamma_{\text{dec}}\Delta_k & \text{if } \rho_k < \eta \end{cases},$$

where ρ_k is defined in (2.5), and where $0 < \gamma_{\text{dec}} < 1 < \gamma_{\text{inc}}$, $\eta \in (0, 1)$, and $\Delta_{\max} > 0$ are fixed parameters.

In our setting, this method cannot be applied directly, since the objective function improvement, $\delta_k := F(x_k) - F(s_k)$, required for the computation of ρ_k, is not known exactly. Instead, we approximate the objective function values using the normalized adaptive multiple importance sampling weights (2.8) for a given sample set $\mathcal{Y}$ by

$$\bar{F}^{\mathcal{Y}}(x) := \sum_{(t_i, y_i) \in \mathcal{Y}} \bar{w}_\sigma(t_i, y_i; x) f(y_i). \tag{2.20}$$

To determine how trustworthy the resulting estimate $\bar{\delta}_k := \bar{F}^{\mathcal{Y}}(x_k) - \bar{F}^{\mathcal{Y}}(s_k)$ is, we assume that a confidence interval $\mathcal{C}_k = [L_k, U_k]$ for $\bar{\delta}_k$ can be computed, with confidence level $1 - \alpha_k$, where $\{\alpha_k\}_{k=0}^\infty$ is a predetermined sequence in $(0, \frac{1}{2})$ with $\sum_{k=0}^\infty \alpha_k < \infty$. At this point, we do not specify the details of how the confidence interval can be estimated. We will only use the fact that $U_k - L_k$ converges to zero as the number of sample points about x_k and s_k goes to infinity. We present in Section 4.1.1 the statistical method for estimating $\mathcal{C}_k$ that we use in our numerical experiments. It does not rely on additional function evaluations.

Given the confidence interval $\mathcal{C}_k = [L_k, U_k]$, the corresponding lower and upper confidence bounds on the ratio of actual decrease in the function to predicted decrease in the model are defined as

$$\rho_k^L = \frac{L_k}{m_k(x_k) - m_k(s_k)} \quad \text{and} \quad \rho_k^U = \frac{U_k}{m_k(x_k) - m_k(s_k)}. \tag{2.21}$$

Choosing fixed constants $\eta_L, \eta_U \in (0,1)$ with $\eta_L < \eta_U$, we then consider the anticipated decrease as significant if

$$\rho_k^L \geq \eta_L, \tag{2.22}$$

and accept the trial point s_k as the new point. Otherwise, if there is high confidence that the ratio is small, i.e.,

$$\rho_k^U \leq \eta_U, \tag{2.23}$$

we conclude that there is significant evidence that there is insufficient improvement in F and reject the trial point s_k. The update conditions are

$$x_{k+1} = \begin{cases} s_k = x_k + p_k & \text{if } \rho_k^L \geq \eta_L \\ x_k & \text{if } \rho_k^L < \eta_L \text{ and } \rho_k^U \leq \eta_U, \end{cases} \tag{2.24a}$$

$$\Delta_{k+1} = \begin{cases} \max\{\Delta_{\text{reset}}, \min\{\gamma_{\text{inc}}\Delta_k, \Delta_{\max}\}\} & \text{if } \rho_k^L \geq \eta_L \\ \gamma_{\text{dec}}\Delta_k & \text{if } \rho_k^L < \eta_L \text{ and } \rho_k^U \leq \eta_U, \end{cases} \tag{2.24b}$$

where $\Delta_{\text{reset}} \in (0, \Delta_{\max}]$ is a constant that denotes the minimum trust region radius for a newly accepted iterate. While somewhat cumbersome, this trust region reset is necessary to prevent the trust region radius from going to zero too quickly and stalling at a non-stationary point. It is possible that both conditions $\rho_k^L \geq \eta_L$ and $\rho_k^U \leq \eta_U$ are satisfied. In our update conditions, we accept the trial point in this case.

If the test is inconclusive (i.e., $\rho_k^L < \eta_L$ and $\rho_k^U > \eta_U$), we augment $\mathcal{Y}$ with additional sample points about both x_k and s_k according to $\phi_\sigma(\cdot\,; x_k)$ and $\phi_\sigma(\cdot\,; s_k)$ for the purpose of improving the accuracy of the estimators $\bar{F}^{\mathcal{Y}}(x_k)$ and $\bar{F}^{\mathcal{Y}}(s_k)$ and shrinking the confidence interval, until we reach a conclusion. At least one of the conditions (2.22) and (2.23) has to be satisfied eventually since $U_k - L_k \to 0$ as the number of samples around x_k and s_k goes to infinity and $\eta_L < \eta_U$.

2.5. Complete Algorithm. We conclude the description of the algorithm with a formal summary.

ALGORITHM 1.

Parameters: *Choose constants* $0 < \eta_L < \eta_U < 1$, $0 < \gamma_{\text{dec}} < 1 < \gamma_{\text{inc}}$, $0 < \Delta_{\text{reset}} < \Delta_{\max}$, $\kappa_{\max} > \kappa_n$, *where* κ_n *is defined in* (2.19).
Let $\{\alpha_k\}$ *be such that* $\alpha_k \in (0, 1/2)$ *for all* k *and* $\sum_{k=0}^{\infty} \alpha_k < \infty$.

Step 0: Initialization.
Set $k \leftarrow 0$ *and select an initial starting point* x_0 *and trust-region radius* Δ_0.
Let $\mathcal{Y}$ *be an initial sample set with* $(n+1)(n+2)/2$ *elements, and set* $\mathcal{Y}_0 \leftarrow \mathcal{Y}$.

Step 1: Model construction. *Build a quadratic model* m_k (2.1) *about* x_k *by solving the regression problem* (2.14).

Step 2: Model improvement. *If* $\operatorname{cond}\left(K_\sigma^{\mathcal{Y}}(x_k)\right) > \kappa_{\max}$, *sample a point* y *according to* $\phi_\sigma(\cdot\,; x_k)$, *let* $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{(x_k, y)\}$, *and go to Step 1.*

Step 3: Compute a trial point. *Find a solution* p_k *to the trust-region subproblem* (2.2) *yielding at least the generalized Cauchy decrease* (2.3) *and let* $s_k := x_k + p_k$.
Sample a point y *according to* $\phi_\sigma(\cdot\,; s_k)$ *and let* $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{(s_k, y)\}$.

Step 4: Evaluate a tentative point. *Compute a* $(1 - \alpha_k)$ *confidence interval* $\mathcal{C}_k$.

Step 4a Acceptance or rejection. *If either* (2.22) *or* (2.23) *holds, update the iterate and trust-region radius according to* (2.24). *Set* $\mathcal{Y}_{k+1} \leftarrow \mathcal{Y}$ *and* $k \leftarrow k+1$, *and go to Step 1.*

Step 4b Confidence interval improvement. *Otherwise draw samples* y^x *and* y^s *according to* $\phi_\sigma(\cdot\,; x_k)$ *and* $\phi_\sigma(\cdot\,; s_k)$, *respectively, augment* $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{(x_k, y^x), (s_k, y^s)\}$, *and return to the beginning of Step 4.*

The algorithm needs to store the set $\mathcal{Y}$ which grows as the algorithm continues. Each element of $\mathcal{Y}$ is a pair (t_i, y_i) of vectors in $\mathbb{R}^n$. Since we assume that n is small, this requires only a moderate amount of memory. For efficiency, an implementation of Algorithm 1 would also store the value $f(y_i)$ of the noisy function alongside (t_i, y_i). Note that the denominator $\phi_\sigma(y_i; t_i)$ in (2.7) remains unchanged and can be stored as well.

Aside from the evaluation of the function f, the main computational effort is the construction of the regression system matrix $K_\sigma^{\mathcal{Y}}(x_k)$, defined in (2.17). This may constitute a significant amount of work when $|\mathcal{Y}|$ is large. Some computational expense can be saved by noting that adding a sample point to $\mathcal{Y}$ corresponds to a rank-1 update of $K_\sigma^{\mathcal{Y}}(x_k)$ when x_k does not change so that low-rank factorization update techniques can be utilized. Recomputing a fresh version that requires the visit of all elements in $\mathcal{Y}$ is only necessary once a new iterate is accepted. Also, when $\|y_i - x_k\| \gg \sigma$, the weight $w_\sigma(t_i, y_i; x_k)$ is essentially zero, and a practical implementation would skip the corresponding element in $\mathcal{Y}$.

To avoid ambiguities in the analysis in the next section, Step 4a takes snapshots of $\mathcal{Y}$ in a sequence $\{\mathcal{Y}_k\}$ which does not need to be stored. Also, some quantities might be "overwritten" during the course of a single iteration. The coefficients (b_k, g_k, H_k) in the definition (2.10) of the model m_k are computed in Step 1. Depending on the outcome of Step 2, this computation might be executed more than once for a given k. In the following analysis, we assume that (b_k, g_k, H_k) corresponds to the model coefficients at the end of an iteration, before k is increased.

3. Algorithm convergence.

In this section we prove the main convergence result for our algorithm, Theorem 3.12, which states that, with probability one (w.p.1), any limit point of the algorithm is a stationary point of F. To carry out the proof, we make the following assumption throughout this section. It implies that the integral in the definition of F in (1.2) is well-defined.

ASSUMPTION 1. *The function f satisfies the growth condition* (1.4).

After setting up the stochastic framework for the analysis in Section 3.1, we establish that the algorithm is well-defined in Section 3.2. Section 3.3 presents the convergence analysis for Algorithm 1.

3.1. Sample Average Approximation.

In this section, we present the tools for the stochastic analysis of the iterates generated by Algorithm 1. Since the samples y_i are drawn at random, the algorithm is a stochastic process, where the iterates x_k as well as the members (t_i, y_i) of the sample sets $\mathcal{Y}_k$ are random variables. (Usually, one would denote random variables with capital letters in this kind of analysis. However, to keep the notation unified, we stay with the lower-case notation.)

For the convergence of Algorithm 1, we need to establish that, w.p.1, the function estimate $\tilde{F}^{\mathcal{Y}_k}$ converges to F *uniformly* in C, even at points $x \in C$ that are not necessarily limit points of the sample means t_i in $\mathcal{Y}_k$. A similar uniform convergence property is required for the regression problem (2.14). The challenge here is that the weights $w_\sigma(t_i, y_i; x)$ are generated by an adaptive multiple importance sampling

procedure. In [49], the authors established results on uniform convergence of sample average approximation with adaptive multiple importance sampling that are the basis for our proof.

Let us first formally describe the stochastic process generated by Algorithm 1. Consider a probability space $(\Omega, \mathcal{F}_\infty, \mathbb{P})$ that supports an infinite sequence $\{Z_i\}$ of iid n-dimensional random vectors that have independent components, each being normally distributed with mean 0 and variance 1. Define $\{\mathcal{F}_i\}$ as the natural filtration of this sequence, i.e., $\mathcal{F}_i$ contains the information in $Z_1, \ldots, Z_i$. The sequence $\{(t_i, y_i)\}$ of sampling means and samples is generated as follows:

ALGORITHM 2.

1. $t_1 = x_0$ *is a given constant. Set* $\mathcal{Y}_0 \leftarrow \emptyset$.
2. *For* $l = 1, 2, 3, \ldots$
   (a) $y_l = t_l + \sigma Z_l$.
   (b) $\mathcal{Y}_l = \mathcal{Y}_{l-1} \cup \{(t_l, y_l)\}$
   (c) t_{l+1} *is computed from* $\mathcal{F}_l$ *in a way expressed by Algorithm 1.*

The iterates t_l in Algorithm 2 are iterates x_k or trial points s_k of Algorithm 1. Each l-iteration in Algorithm 2 corresponds to a single sample generation in Algorithm 1. Therefore one k-iteration of Algorithm 1 corresponds to several l-iterations in Algorithm 2 and we add a single sample point per l-iteration to $\mathcal{Y}_l$, while the sample sets $\mathcal{Y}_k$ in Algorithm 1 can grow by more than one element per k-iteration. Consequently, $\{\mathcal{Y}_k\}$ is a subsequence of $\{\mathcal{Y}_l\}$.

Each y_l is conditionally normal with mean vector t_l and covariance matrix $\sigma^2 I$ given $\mathcal{F}_{l-1}$. The density of this distribution is given by $\phi_\sigma(\,\cdot\,; t_l)$. The analysis in [49] applies to Algorithm 2 since it permits that the random variables t_l are conditioned on the history of the previous iterations. The results in [49] require that the sequence $\{t_l\}$ is contained in a compact set C; this is ensured due to (2.2c).

The first result from [49] addresses the uniform convergence of function estimates.

THEOREM 3.1. *Let* $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ *be a function for which there exist constants* $K_g > 0$ *and* $a_g > 0$ *such that, for all* $y \in \mathbb{R}^n$, $|g(y)| \leq K_g \exp(a_g \|y\|)$, *and let* $C \subset \mathbb{R}^n$ *be a compact set. Then, w.p.1, for any* $\epsilon > 0$ *there exists* $L_\epsilon > 0$ *such that*

$$\forall l > L_\epsilon, \ \forall x \in C, \quad \left| \frac{1}{l} \sum_{i=1}^{l} w_\sigma(t_i, y_i; x) g(y_i) - \int g(y) \phi_\sigma(y; x) \, \mathrm{d}y \right| < \epsilon.$$

Proof. This result is an application of Theorem 1 in [49] which proves uniform convergence of sample average approximations. The symbols x, ξ, $F(\xi)$, $h(x, \xi)$, and $\phi_i(\xi_i)$ in [49] correspond to the symbols x, y, $g(y)$, $\phi_\sigma(y, x)$, and $\phi_\sigma(y_i, t_i)$ here. Proposition 1 in [49] shows that the assumptions for Theorem 1 in [49] are satisfied because all distributions are normal with fixed variance and bounded mean. Note that Assumption 9 for Proposition 1 in [49] requires that g is differentiable w.r.t. x, which is vacuous since g does not depend on x. ∎

The second result establishes convergence of the minimizers of a parametrized sample average approximation of an optimization problem. Let $\{\hat{x}_l\}$ be a random sequence in a compact set $C \subset \mathbb{R}^n$ converging to a random limit point $\hat{x}_*$, and let $h : \mathbb{R}^d \times \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}$ be a measurable function. With this, we define the optimization problem

$$\min_{\zeta \in \mathbb{R}^d} \Theta_*(\zeta) = \int h(\zeta, \hat{x}_*, y) \phi_\sigma(y; \hat{x}_*) \, \mathrm{d}y \tag{3.1}$$

together with a sequence of sample average approximation problems ($l = 1, 2, 3, \ldots$)

$$\min_{\zeta \in \mathbb{R}^d} \Theta_l(\zeta) = \frac{1}{l} \sum_{i=1}^{l} w_\sigma(t_i, y_i; \hat{x}_l) h(\zeta, \hat{x}_l, y_i). \tag{3.2}$$

Note that the objective function in (3.2) depends on the sequence $\{\hat{x}_l\}$. We denote the solution sets of (3.1) and (3.2) by S_* and S_l, respectively. The sequence $\{\hat{x}_l\}$ does not have to correspond to (a renumbering of) the sequence of iterates $\{x_k\}$ in Algorithm 1.

THEOREM 3.2. *Let $\mathcal{Z} \subset \mathbb{R}^d$ and $C \subset \mathbb{R}^n$ be compact sets, and let $h : \mathbb{R}^d \times \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}$ be a function with the following properties:*

(i) For each $y \in \mathbb{R}^n$, $h(\zeta, x, y)$ is differentiable in (ζ, x); and

(ii) there exist $K_h > 0$ and $a_h > 0$ so that for all $\zeta \in \mathcal{Z}$, $x \in C$, and $y \in \mathbb{R}^n$

$$|h(\zeta, x, y)| \leq K_h \exp(a_h \|y\|)$$
$$\|\nabla_{\zeta,x} h(\zeta, x, y)\| \leq K_h \exp(a_h \|y\|),$$

(iii) h is bounded below.

Let $\{\hat{x}_l\}$ be a random sequence in C that converges to a random limit point $\hat{x}_* \in C$. Assume that, w.p.1, $S_* = \{\zeta_*\}$ is a nonempty singleton and contained in $\mathcal{Z}$, and, for l large enough, $S_l = \{\zeta_l\}$ is a nonempty singleton and contained in $\mathcal{Z}$. Then $\lim_{l\to\infty} \zeta_l = \zeta_*$ w.p.1.

Proof. This result is an application of Theorem 4 in [49] which proves convergence of the optimal solutions of sample average approximation problems in the presence of converging parameters ($\hat{x}_l$ in our case). Again, Proposition 1 in [49] shows that Assumptions 1-3 for Theorem 4 in [49] are satisfied because all distributions are normal with fixed variance and bounded mean. Here, Assumption 9 for Proposition 1 in [49] corresponds to the assumptions (i) and (ii) above. Theorem 4 in [49] also requires that $G(\zeta, x, y) = h(\zeta, x, y)\phi_\sigma(y; x)$ is bounded below which is implied by (iii) and the fact that $\phi_\sigma(\,\cdot\,;\,\cdot\,)$ is positive and bounded. ∎

3.2. Well-Definedness. In Step 2 of Algorithm 1, sample points drawn from $\phi_\sigma(.; x_k)$ are added to $\mathcal{Y}$ until the condition number of $K_\sigma^{\mathcal{Y}}(x_k)$ is at most $\kappa_{\max}$. To show well-definedness of the algorithm, we need to prove that this can be achieved in a finite number of samples.

LEMMA 3.3. *The condition number of the $(n+1) \times (n+1)$ matrix*

$$A_n = \begin{bmatrix} 1 & \frac{1}{2} & \cdots & \cdots & \frac{1}{2} \\ \frac{1}{2} & \frac{3}{4} & \frac{1}{4} & \cdots & \frac{1}{4} \\ \vdots & \frac{1}{4} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{4} & \cdots & \frac{1}{4} & \frac{3}{4} \end{bmatrix} \tag{3.3}$$

is κ_n given in (2.19).

This result is proved in [28] by recursively applying the Laplace expansion to the determinant of the characteristic polynomial of A_n.

LEMMA 3.4. *For every $\epsilon > 0$, there exists, w.p.1, an integer $L_\epsilon > 0$ and a number $K_K > 0$ so that $\|K_\sigma^{\mathcal{Y}_l}(x)\| \geq K_K$ and $\left|\text{cond}\left(K_\sigma^{\mathcal{Y}_l}(x)\right) - \kappa_n\right| < \epsilon$ for all $x \in C$ and $l > L_\epsilon$, where κ_n is given in (2.19).*

Proof. Let $x \in C$. Recalling (2.17), we are interested in the condition number of the matrix $K_\sigma^{\mathcal{Y}_l}(x) = M_l(x)^T W_l(x) M_l(x)$, where the ith row of $M_l(x) \in \mathbb{R}^{l \times \frac{1+n(n+1)}{2}}$ is defined as

$$[M_l(x)]_i = \left(1, \tfrac{z_{i1}^2}{2}, \ldots, \tfrac{z_{in}^2}{2}, z_{i1}, \ldots, z_{in}, z_{i1}z_{i2}, \ldots, z_{i1}z_{in}, z_{i2}z_{i1}, z_{i2}z_{i3}, \ldots, z_{i(n-1)}z_{in}\right)$$

with $z_{ij} = (y_i^{(j)} - x^{(j)})/\sigma$ (note that w.l.o.g. we reordered the elements compared to (2.16)) and where $W_l(x) \in \mathbb{R}^{l \times l}$ is a diagonal matrix with the elements $w_i = w_\sigma(t_i, y_i; x)/l$ on its diagonal. We then have

$$K_\sigma^{\mathcal{Y}_l}(x) = \sum_{i=1}^{l} w_i [M_l(x)]_i^T [M_l(x)]_i \tag{3.4}$$

$$= \sum_{i=1}^{l} \frac{w_\sigma(t_i, y_i; x)}{l} \begin{bmatrix} 1 & \frac{z_{i1}^2}{2} & \cdots & \frac{z_{in}^2}{2} & z_{i1} & z_{i2} & \cdots & z_{i(n-1)}z_{in} \\ \frac{z_{i1}^2}{2} & \frac{z_{i1}^4}{4} & \cdots & \frac{z_{i1}^2 z_{in}^2}{4} & \frac{z_{i1}^3}{2} & \frac{z_{i1}^2 z_{i2}}{2} & \cdots & \frac{z_{i1}^2 z_{i(n-1)} z_{in}}{2} \\ \vdots & & \ddots & \vdots & \vdots & \vdots & & \vdots \\ \frac{z_{in}^2}{2} & \frac{z_{in}^2 z_{i1}^2}{4} & \cdots & \frac{z_{in}^4}{4} & \frac{z_{in}^2 z_{i1}}{2} & \frac{z_{in}^2 z_{i2}}{2} & \cdots & \frac{z_{i(n-1)} z_{in}^3}{2} \\ z_{i1} & \frac{z_{i1}^3}{2} & & & z_{i1}^2 & z_{i1}z_{i2} & \cdots & z_{i1}z_{i(n-1)}z_{in} \\ z_{i2} & \frac{z_{i1}^2 z_{i2}}{2} & & & z_{i1}z_{i2} & z_{i2}^2 & \cdots & z_{i2}z_{i(n-1)}z_{in} \\ \vdots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ z_{i(n-1)}z_{in} & \vdots & & & \vdots & \vdots & & z_{i(n-1)}^2 z_{in}^2 \end{bmatrix}.$$

We now apply Theorem 3.1 to each component of the matrix $K_\sigma^{\mathcal{Y}_l}(x)$. Let $g(z)$ be the function for computing a given matrix component in (3.4); for example, $g(z) = \frac{z_{i(n-1)}z_{in}}{2}$ for the last entry in the first row. Using Theorem 3.1 we see that, w.p.1, the corresponding entry in $K_\sigma^{\mathcal{Y}_l}(x)$ converges uniformly over C to

$$\int g\left(\tfrac{y-x}{\sigma}\right) \phi_\sigma(y; x) \mathrm{d}y = \int g(z) \phi_1(z; 0) \mathrm{d}z = \mathbb{E}[g(Z)]$$

where $Z \sim \mathcal{N}(0, I)$. Using the fact that, for $j_1, j_2, j_3, j_4 \in \{1, \ldots, n\}$ mutually different, $\mathbb{E}[Z_{j_1}^2] = 1$, $\mathbb{E}[Z_{j_1}^4] = 3$, $\mathbb{E}[Z_{j_1}^2 Z_{j_2}^2] = 1$, $\mathbb{E}[Z_{j_1}] = 0$, $\mathbb{E}[Z_{j_1} Z_{j_2}] = 0$, $\mathbb{E}[Z_{j_1} Z_{j_2}^2] = 0$, $\mathbb{E}[Z_{j_1}^3] = 0$, $\mathbb{E}[Z_{j_1} Z_{j_2} Z_{j_3}] = 0$, $\mathbb{E}[Z_{j_1}^2 Z_{j_2} Z_{j_3}] = 0$, we have that $K_\sigma^{\mathcal{Y}_l}(x)$ converges uniformly over C to $\tilde{A} = \begin{bmatrix} A_n & 0 \\ 0 & I \end{bmatrix}$ w.p.1, where A_n is defined in (3.3). The claim then follows with $K_K = \|\tilde{A}\|/2$ from the continuity of the norm and of the condition number and Lemma 3.3. ∎

COROLLARY 3.5. *W.p.1, in each iteration k, Step 2 of Algorithm 1 terminates after adding a finite number of samples.*

Proof. By the previous lemma, adding sample points eventually yields $\operatorname{cond}\left(K_\sigma^{\mathcal{Y}}(x_k)\right) \leq \kappa_{\max}$, because $\kappa_{\max} > \kappa_n$ and $x_k \in C$. ∎

3.3. Convergence Proof. We start with an auxiliary results that establishes that the right-hand side in (2.18) is bounded.

LEMMA 3.6. *W.p.1, there exists a bound $K_r > 0$ so that $\|r_\sigma^{\mathcal{Y}_k}(x_k)\| \leq K_r$ for k sufficiently large, where $r_\sigma^{\mathcal{Y}_k}(x_k)$ is defined in (2.17).*

Proof. Let $x \in C$ and, in analogy to (2.17), define $r_\sigma^{\mathcal{Y}_l}(x) = M_l(x)^T W_l(x) f(\mathcal{Y}_l)$ with $M_l(x)$ and $W_l(x)$ defined in the proof of Lemma 3.4. Let $r_l(x)$ be a component

of the vector $r_\sigma^{\mathcal{Y}_l}(x)$. It is of the form $r_l(x) = \frac{1}{l}\sum_{i=1}^{l} q\left(\frac{y_i - x}{\sigma}\right) w_\sigma(t_i, y_i; x) f(y_i)$, where $q(\,\cdot\,)$ is a polynomial of degree at most two. Setting $g(y) = q\left(\frac{y-x}{\sigma}\right) f(y)$, Theorem 3.1 then implies that, w.p.1, $r_l(x)$ converges uniformly over C to $Q(x) = \int q\left(\frac{y-x}{\sigma}\right) f(y)\phi_\sigma(y; x)\mathrm{d}y$ as $l \to \infty$. Because C is a compact set, $Q(x)$ is uniformly bounded by, say, $\tilde{K}_q$. Since $\{\mathcal{Y}_k\}$ is a subsequence of $\{\mathcal{Y}_l\}$, uniform convergence implies that, w.p.1, there exists an integer $N_q > 0$ so that for all $k > N_q$ and all $x \in C$ we have $r_k(x) \le K_q := \tilde{K}_q + 1$, and in particular, $r_k(x_k) \le K_q$ for all $k > N_q$.

The claim of the lemma follows from the fact that this argument can be conducted for each of the finitely many vector components in $r_\sigma^{\mathcal{Y}_l}(x)$. ∎

Next we show that the regression problem (2.14) indeed recovers the gradient and Hessian of F in the limit.

LEMMA 3.7. *Suppose Assumption 1 holds and let $\{x_{\varphi(k)}\}$ be a subsequence of the (random) sequence of iterates $\{x_k\}$ generated by Algorithm 1 converging to a (random) limit point x_*. Then, w.p.1, $\lim_{k\to\infty} g_{\varphi(k)} = \nabla F(x_*)$ and $\lim_{k\to\infty} H_{\varphi(k)} = \nabla^2 F(x_*)$.*

Proof. Without loss of generality we assume that $\varphi(k) = k$. We apply Theorem 3.2 to the regression problem (2.11). Its approximation (2.14) corresponds to (3.2), where, recalling $\zeta = (b, g, H)$,

$$h(\zeta, x, y) = (m_\zeta(y; x) - f(y))^2 = \left(b + g^T(y - x) + \frac{1}{2}(y - x)^T H(y - x) - f(y)\right)^2.$$

From Lemma 2.1, we know that S_* is a singleton, say $S_* = \{\zeta_*\}$. Furthermore, by Step 2 in Algorithm 1, $\operatorname{cond}\left(K_\sigma^{\mathcal{Y}}(x_k)\right) \le \kappa_{\max}$, so the set S_k is also a singleton for any k, say $S_k = \{\zeta_k\}$, with $\zeta_k = (\hat{b}_k,\, \sigma\hat{g}_k,\, \sigma^2\hat{H}_k)$ where $\hat{\zeta}_k = (\hat{b}_k, \hat{g}_k, \hat{H}_k)$ is the solution of the linear system (2.18). Recall that for a nonsingular matrix A it is $\operatorname{cond}(A) = \|A\| \cdot \|A^{-1}\|$. From Lemmas 3.4 and 3.6, we have w.p.1 that $\|K_\sigma^{\mathcal{Y}}(x_k)\| \ge K_K$ and $\|r_\sigma^{\mathcal{Y}}(x_k)\| \le K_r$ for k sufficiently large, and obtain $\|\hat{\zeta}_k\| = \left\|\left(K_\sigma^{\mathcal{Y}}(x_k)\right)^{-1} r_\sigma^{\mathcal{Y}}(x_k)\right\| \le \left\|\left(K_\sigma^{\mathcal{Y}}(x_k)\right)^{-1}\right\| \|r_\sigma^{\mathcal{Y}}(x_k)\| \le \frac{\operatorname{cond}\left(K_\sigma^{\mathcal{Y}}(x_k)\right)}{\|K_\sigma^{\mathcal{Y}}(x_k)\|} K_r \le \frac{\kappa_{\max} K_r}{K_K}$. Therefore,

$$\|\zeta_k\| = \left\| \begin{pmatrix} \hat{b}_k \\ \sigma\hat{g}_k \\ \sigma^2 \operatorname{vect}(\hat{H}_k) \end{pmatrix} \right\| \le \max\{1, \sigma^2\} \left\| \begin{pmatrix} \hat{b}_k \\ \hat{g}_k \\ \operatorname{vect}(\hat{H}_k) \end{pmatrix} \right\| \le \max\{1, \sigma^2\} \frac{\kappa_{\max} K_r}{K_K},$$

where $\operatorname{vect}(\hat{H}_k)$ is the vector consisting of the n^2 matrix elements of $\hat{H}_k$. Then $\mathcal{Z} = \{\zeta : \|\zeta\| \le \max\{\|\zeta_*\|, \max\{1, \sigma^2\}\frac{\kappa_{\max} K_r}{K_K}\}$ is a compact set containing S_* and S_k for sufficiently large k.

We can now conclude the proof by applying Theorem 3.2. It remains to show that conditions (i) and (ii) in Theorem 3.2 hold. For a fixed $y \in \mathbb{R}^n$, h is a polynomial in (ζ, x) and therefore differentiable in (ζ, x), yielding condition (i). Furthermore, since $\mathcal{Z}$ and C are compact, there exists $K_y > 0$ so that for all $\zeta \in \mathcal{Z}$, $x \in C$, and $y \in \mathbb{R}^n$ we have $|h(\zeta, x, y)| \le K_y \max\{|f(y)|, f(y)^2\}$ and $\|\nabla_{\zeta,x} h(\zeta, x, y)\| \le K_y |f(y)|$. This, together with Assumption 1, implies condition (ii). ∎

Now we show that, for any convergent sequence $\{\hat{x}_k\}$, the function approximations using self-normalized weights, $\bar{F}^{\mathcal{Y}}(\hat{x}_k)$, defined in (2.20), converge to the true value of F at the limit point. This sanity check justifies the use of $\bar{F}^{\mathcal{Y}}$ in the step acceptance criterion.

LEMMA 3.8. *Suppose Assumption 1 holds and let $\{\hat{x}_k\}$ be any random sequence in C converging to a random limit point $\hat{x}_*$. Then, w.p.1, $\lim_{k\to\infty} \bar{F}^{\mathcal{Y}_k}(\hat{x}_k) = F(\hat{x}_*)$.*

Proof. To use the theorems in Section 3.1, we expand the sequence $\{\hat{x}_k\}$ to a sequence $\{\hat{x}_l\}$ by appropriately repeating elements.

Following similar arguments as in the proof of Lemma 2.1, one can see that $F(\hat{x}_*) = \zeta_* := \operatorname{argmin}_{\zeta\in\mathbb{R}} \int (f(y)-\zeta)^2 \phi_\sigma(y;\hat{x}_*)\mathrm{d}y$. Now consider the corresponding sequence of SAA problems

$$\min_{\zeta} \frac{1}{l}\sum_{i=1}^{l} w_\sigma(t_i,y_i;\hat{x}_l)\left(f(y_i)-\zeta\right)^2 \tag{3.5}$$

with solutions ζ_l. The first order optimality condition yields $\sum_{i=1}^{l} w_\sigma(t_i,y_i;\hat{x}_l)(f(y_i)-\zeta_l)=0$, which shows that

$$\zeta_l = \sum_{i=1}^{l} \frac{w_\sigma(t_i,y_i;\hat{x}_l)}{\sum_{j=1}^{l} w_\sigma(t_j,y_j;\hat{x}_l)} f(y_i) = \bar{F}^{\mathcal{Y}_l}(\hat{x}_l) \tag{3.6}$$

is the optimal solution of (3.5) for a given l (recall (2.8) and (2.20)).

Now, applying Theorem 3.1 twice with $g(y)=1$ and $g(y)=f(y)$, we obtain that, w.p.1, there exists $L>0$ so that for all $l>L$ and all $x\in C$

$$\left|\frac{1}{l}\sum_{i=1}^{l} w_\sigma(t_i,y_i;x)-1\right| \le \frac{1}{2} \quad \text{and} \quad \left|\tilde{F}^{\mathcal{Y}_l}(x)-F(x)\right| \le 1,$$

where $\tilde{F}^{\mathcal{Y}_l}$ is the non-normalized approximation (2.7). Here, we used the fact that the integral of the pdf ϕ_σ is 1. This, together with (3.6) and (2.7), implies that for $l>L$

$$|\zeta_l| = \left|\frac{1}{\frac{1}{l}\sum_{j=1}^{l} w_\sigma(t_j,y_j;\hat{x}_l)}\tilde{F}^{\mathcal{Y}_l}(\hat{x}_l)\right| \le 2(1+F_{\max}),$$

where $F_{\max}=\max\{F(x): x\in C\}$. The claim then follow from the application of Theorem 3.2 to (3.5) with the function $h(\zeta,\hat{x}_l,y)=(\zeta-f(y))^2$ and with $\mathcal{Z}=\{\zeta\in\mathbb{R}: |\zeta|\le 2(1+F_{\max})\}$. ∎

The next lemma shows that we can bound the Hessian approximation and get a uniform bound on the gradient error in a neighborhood of x_*.

LEMMA 3.9. *Suppose Assumption 1 holds and let $\{x_{\varphi(k)}\}$ be a subsequence of $\{x_k\}$ converging to a limit point x_*. Then there exists $\beta>0$ such that, w.p.1, $\|H_{\varphi(k)}\|\le\beta$ for all k. Furthermore, w.p.1, $\lim_{k\to\infty}\|g_{\varphi(k)}-\nabla F\left(x_{\varphi(k)}\right)\|=0$.*

Proof. Note first that Lemma 3.7 guarantees that, w.p.1, $H_{\varphi(k)}\to\nabla^2 F(x_*)$ and the sequence is thus bounded, which proves the first claim.

For any k we have

$$\begin{aligned}\|g_{\varphi(k)}-\nabla F\left(x_{\varphi(k)}\right)\| &\le \|g_{\varphi(k)}-\nabla F(x_*)\| + \|\nabla F\left(x_{\varphi(k)}\right)-\nabla F(x_*)\| \\ &\le \|g_{\varphi(k)}-\nabla F(x_*)\| + \tilde{\beta}\|x_{\varphi(k)}-x_*\|,\end{aligned}$$

where $\tilde{\beta}$ is a uniform upper bound of $\|\nabla^2 F(x)\|$ for all x in the compact set C. The first term converges to zero, w.p.1, by Lemma 3.7, and the second term converges to zero because x_* is the limit point of $\{x_{\varphi(k)}\}$. ∎

The next lemma establishes that the trust region radius is bounded away from zero if a subsequence of the iterates approaches a non-stationary limit point. Recall that $\chi(x) = \tilde{\chi}(\nabla F(x), x)$ with $\tilde{\chi}$ defined in (2.4).

LEMMA 3.10. *Suppose Assumption 1 holds and let $\{x_{\varphi(k)}\}$ be a subsequence of $\{x_k\}$ converging to a limit point x_*. If $\chi(x_*) > 0$, then, w.p.1, we have $\liminf_k \Delta_{\varphi(k)} > 0$.*

Proof. Suppose without loss of generality that $\{x_{\varphi(k)}\}$ is such that $x_{\varphi(k)-1}$ is also in the subsequence if $x_{\varphi(k)-1} = x_{\varphi(k)}$. In other words, if $x_{\varphi(k)}$ is in the subsequence, then iterates from all preceding iterations that rejected the trial point and stayed at the same point $x_{\varphi(k)}$ are also in the subsequence.

Let $\varepsilon := \chi(x_*)/2 > 0$. From (2.5) we have

$$\left|\rho_{\varphi(k)} - 1\right| = \left| \frac{\left[F(x_{\varphi(k)}) - F(s_{\varphi(k)})\right] - \left[m_k(x_{\varphi(k)}) - m_k(s_{\varphi(k)})\right]}{m_{\varphi(k)}(x_{\varphi(k)}) - m_{\varphi(k)}(s_{\varphi(k)})} \right|.$$

The numerator is the difference between the actual decrease and the predicted decrease, and, using (2.1), Taylor's theorem, and (2.2b), can be bounded, w.p.1, in the following manner:

$$\begin{aligned}
&\left|\left[F(x_{\varphi(k)}) - F(s_{\varphi(k)})\right] - \left[m_{\varphi(k)}(x_{\varphi(k)}) - m_{\varphi(k)}(s_{\varphi(k)})\right]\right| \\
=&\left| g_{\varphi(k)}^T p_{\varphi(k)} + \tfrac{1}{2} p_{\varphi(k)}^T H_{\varphi(k)} p_{\varphi(k)} - \left[F(x_{\varphi(k)} + p_{\varphi(k)}) - F(x_{\varphi(k)})\right] \right| \\
=&\Bigg| g_{\varphi(k)}^T p_{\varphi(k)} + \tfrac{1}{2} p_{\varphi(k)}^T H_{\varphi(k)} p_{\varphi(k)} - \nabla F(x_{\varphi(k)})^T p_{\varphi(k)} - \\
&\qquad \int_0^1 \left(\nabla F(x_{\varphi(k)} + t p_{\varphi(k)}) - \nabla F(x_{\varphi(k)})\right)^T p_{\varphi(k)} \mathrm{d}t \Bigg| \\
\leq& \left\| g_{\varphi(k)} - \nabla F(x_{\varphi(k)}) \right\| \|p_{\varphi(k)}\| + \left| \tfrac{1}{2} p_{\varphi(k)}^T H_{\varphi(k)} p_{\varphi(k)} \right| + \int_0^1 \tilde{\beta} \|t p_{\varphi(k)}\| \, \|p_{\varphi(k)}\| \mathrm{d}t \\
\leq& \left\| g_{\varphi(k)} - \nabla F(x_{\varphi(k)}) \right\| \|p_{\varphi(k)}\| + \tfrac{1}{2} (\|H_{\varphi(k)}\| + \tilde{\beta}) \|p_{\varphi(k)}\|^2 \\
\leq& \left\| e_{\varphi(k)} \right\| \|p_{\varphi(k)}\| + \tilde{\beta} \|p_{\varphi(k)}\|^2 \leq \left\| e_{\varphi(k)} \right\| \Delta_{\varphi(k)} + \tilde{\beta} \Delta_{\varphi(k)}^2,
\end{aligned} \tag{3.7}$$

where $e_{\varphi(k)} := \|g_{\varphi(k)} - \nabla F(x_{\varphi(k)})\|$, and $\tilde{\beta} \geq \beta$ is a uniform upper bound of $\|\nabla^2 F(x)\|$ for $x \in C$, with β from Lemma 3.9.

Similarly, the denominator is the predicted decrease obtained by the model over the trust region, which we require to be greater than the generalized Cauchy decrease $\xi_{\varphi(k)}$ defined in (2.3). Because the function $\tilde{\chi}$ defined in (2.4) is continuous, Lemma 3.9 implies that there exists, w.p.1, $K_1 > 0$ such that for any $k \geq K_1$ we have $\chi_{\varphi(k)} = \tilde{\chi}(g_{\varphi(k)}, x_{\varphi(k)}) \geq \varepsilon$ since $\chi(x_*) = \tilde{\chi}(\nabla F(x_*), x_*)$. For $k \geq K_1$, we then have with (3.7), (2.3), and Lemma 3.9 that, w.p.1,

$$|\rho_{\varphi(k)} - 1| \leq \frac{\|e_{\varphi(k)}\| \Delta_{\varphi(k)} + \tilde{\beta} \Delta_{\varphi(k)}^2}{\frac{1}{2} \varepsilon \min\left\{\frac{\varepsilon}{\tilde{\beta}}, \Delta_{\varphi(k)}\right\}}. \tag{3.8}$$

Let $\gamma := \frac{(1-\eta_U)\varepsilon}{8}$ with η_U from (2.24). By Lemma 3.9, w.p.1 there exists K_2 such that for $k \geq K_2$, $\|e_{\varphi(k)}\| < \gamma$. Furthermore, for $\Delta_{\varphi(k)} \leq \tilde{\Delta} := \min\left\{\Delta_{\text{reset}}, \frac{(1-\eta_U)\varepsilon}{8\tilde{\beta}}\right\} \leq$

$\frac{\epsilon}{\beta}$ we have

$$\frac{\gamma\Delta_{\varphi(k)} + \tilde{\beta}\Delta_{\varphi(k)}^2}{\frac{1}{2}\varepsilon \min\left\{\frac{\varepsilon}{\beta}, \Delta_{\varphi(k)}\right\}} \leq \frac{2\gamma}{\varepsilon} + \frac{2\tilde{\beta}}{\varepsilon}\Delta_{\varphi(k)} \leq \frac{1-\eta_U}{2} < 1 - \eta_U.$$

With (3.8) this yields that for $k \geq K := \max\{K_1, K_2\}$, we have $|\rho_{\varphi(k)} - 1| < (1 - \eta_U)$ whenever $\Delta_{\varphi(k)} \leq \tilde{\Delta}$, which implies that

$$\rho_{\varphi(k)} > \eta_U \quad \text{whenever} \quad \Delta_{\varphi(k)} \leq \tilde{\Delta}. \tag{3.9}$$

For every iteration k, define the event $A_k := \left\{\Delta_{\varphi(k)} \leq \tilde{\Delta} \text{ and } s_{\varphi(k)} \text{ is rejected}\right\}$, in which case a trial point is erroneously rejected. Suppose A_k occurs in an iteration $k \geq K$. Rejection of the trial step $s_{\varphi(k)}$ entails $\rho_{\varphi(k)}^U \leq \eta_U$. Together with (2.21) and (3.9), this implies that

$$\frac{U_{\varphi(k)}}{m_{\varphi(k)}(x_{\varphi(k)}) - m_{\varphi(k)}(s_{\varphi(k)})} = \rho_{\varphi(k)}^U \leq \eta_U < \rho_{\varphi(k)} = \frac{F(x_{\varphi(k)}) - F(s_{\varphi(k)})}{m_{\varphi(k)}(x_{\varphi(k)}) - m_{\varphi(k)}(s_{\varphi(k)})}$$

and hence $U_{\varphi(k)} < F(x_{\varphi(k)}) - F(s_{\varphi(k)}) = \delta_{\varphi(k)}$. By the definition of the $(1 - \alpha_{\varphi(k)})$-confidence interval $\mathcal{C}_{\varphi(k)} = [L_{\varphi(k)}, U_{\varphi(k)}]$, we have $\mathbb{P}[U_{\varphi(k)} < \delta_{\varphi(k)}] \leq \alpha_{\varphi(k)}$. Therefore, $\mathbb{P}[A_k] \leq \alpha_{\varphi(k)}$. Since the sequence $\{\alpha_k\}$ has been chosen so that $\sum_k \alpha_k < \infty$, it follows from the Borel-Cantelli Theorem [24] that $\mathbb{P}[A_k \text{ infinitely often } (i.o.)] = 0$. Therefore, there exists, w.p.1, a K' such that for all $k \geq K'$, at iteration $\varphi(k)$, either $\Delta_{\varphi(k)} > \tilde{\Delta}$ or $s_{\varphi(k)}$ is accepted.

For the purpose of deriving a contradiction, suppose that $\Delta_{\varphi(k)} < \gamma_{\text{dec}}\tilde{\Delta}$ for some $k > K'$. By the choice of K', $s_{\varphi(k)}$ must be accepted. If the previous trial point $s_{\varphi(k)-1}$ had also been accepted then the update rule (2.24b) yields $\Delta_{\varphi(k)} \geq \Delta_{\text{reset}} \geq \tilde{\Delta} > \gamma_{\text{dec}}\tilde{\Delta} > \Delta_{\varphi(k)}$, a contradiction. On the other hand, if $s_{\varphi(k)-1}$ had been rejected at iterate $x_{\varphi(k)-1}$, we had assumed without loss of generality that then also $x_{\varphi(k)-1}$ is in the subsequence, i.e., $x_{\varphi(k-1)} = x_{\varphi(k)-1}$. Therefore, $s_{\varphi(k-1)} = s_{\varphi(k)-1}$, and we would have $\Delta_{\varphi(k-1)} = \frac{1}{\gamma_{\text{dec}}}\Delta_{\varphi(k)} < \tilde{\Delta}$. This, however, is in contradiction to the fact that event A_{k-1} does not occur for $k - 1 \geq K'$.

Overall, we have shown that $\Delta_{\varphi(k)} \geq \gamma_{\text{dec}}\tilde{\Delta}$ for all $k > K'$. ∎

The final lemma before the main theorem shows that, eventually, points are accepted only if the reduction condition is satisfied for the exact function F.

LEMMA 3.11. *Suppose Assumption 1 holds. Then, w.p.1, we have for sufficiently large k, that $F(x_k) - F(x_{k+1}) \geq \eta_L\,(m_k(x_k) - m_k(x_{k+1}))$ if $x_{k+1} = s_k$.*

Proof. Define the event $B_k := \{(2.22) \text{ holds and } \rho_k < \eta_L\}$. It is true if the algorithm accepts a trial point based on function estimates that incorrectly predict sufficient decrease in the true function values. Similar to the arguments in the previous proof, the definition of the confidence interval $\mathcal{C}_k$ yields $\mathbb{P}[L_k > \delta_k] \leq \alpha_k$, which implies $\mathbb{P}[B_k] \leq \alpha_k$. Since the sequence $\{\alpha_k\}$ has been chosen so that $\sum_k \alpha_k < \infty$, it follows from the Borel-Cantelli Theorem [24] that $\mathbb{P}[B_k,\ i.o.] = 0$. Therefore, w.p.1, for sufficiently large k it is $\rho_k \geq \eta_L$ whenever a trial point is accepted. ∎

We can finally state the main theorem.

THEOREM 3.12. *Suppose Assumption 1 holds, and let $\{x_k\}$ be the (random) sequence of iterates generated by Algorithm 1. Then, w.p. 1, we have $\chi(x_*) = 0$ for any (random) limit point x_* of $\{x_k\}$.*

Proof. Let x_* be a limit point of $\{x_k\}$ and $\{x_{\varphi(k)}\}$ be a subsequence of $\{x_k\}$ converging to x_*. For the purpose of deriving a contradiction, suppose that $\chi(x_*) > 0$.

First note that there must be, w.p.1, an infinite number of successful iterations, since otherwise $\lim_k \Delta_k = 0$, in contradiction to Lemma 3.10. Without loss of generality, we may rearrange $\varphi(k)$ such that $x_{\varphi(k)+1} = s_{\varphi(k)}$, i.e., we renumber the subsequence counter to include only the successful iterations.

The continuity of $\tilde{\chi}$ and Lemmas 3.9 and 3.10 guarantee that there exists, w.p.1, $K_1 > 0$ such that for any $k \geq K_1$ we have $\chi_{\varphi(k)} = \tilde{\chi}(g_{\varphi(k)}, x_{\varphi(k)}) \geq \varepsilon := \frac{\chi(x_*)}{2}$ and $\Delta_{\varphi(k)} \geq \Delta_L$ for some $\Delta_L > 0$. Lemma 3.11 shows that, w.p.1, the sequence $\{F(x_k)\}$ is non-increasing for large k, since either $x_{k+1} = x_k$ or $F(x_k) - F(x_{k+1}) \geq \eta_L(m_k(x_k) - m_k(s_k)) \geq \eta_L \xi_k > 0$ by (2.3). Again applying Lemma 3.11 and (2.3) to the subsequence, we obtain, w.p.1, $F(x_{\varphi(k)}) - F(x_{\varphi(k+1)}) \geq \frac{1}{2}\eta_L \varepsilon \min\left\{\frac{\varepsilon}{\beta}, \Delta_L\right\} > 0$ for all $k \geq K_1$ with sufficiently large K_1, where β is the constant in Lemma 3.9. Because $F(x)$ is bounded below on the compact set C containing the iterates, this is not possible, and we conclude that $\chi(x_*) = 0$. □

4. Numerical experiments. In this section we examine the numerical performance of Algorithm 1. Since we motivated the objective function (1.2) as a new way to handle computational noise (see Section 1.1), our experiments are done with test problems that emulate such noise [32]. As baseline for our comparison we use the basic version of the trust-region DFO algorithm [3, 21, 44] which Algorithm 1 is based on; it is described below in Section 4.2. This method is applied to the minimization of the original function f instead of the smoothed function F since the latter cannot be computed directly.

The purpose of these experiments is to demonstrate that the proposed algorithm converges with a comparable amount of effort and provides superior solutions. We show that it reaches, in the early stage of the optimization, good objective values within a similar number of function evaluations. We also compare the quality of the final solutions computed by the two methods. Here we take the view that we want to explore whether our modifications of the baseline algorithm overcome the tendency of the original algorithm to get trapped in artificial local minima of the noisy function f. Since the algorithms seek to minimize different objective functions, F versus f, a direct comparison is somewhat ill-defined. For simplicity, in the experiments we use the value of the original noiseless smooth function $\bar{f}$ (from which f is constructed) to measure solution quality, but we declare that values within the noise level are equally good.

Alternative baseline algorithms are conceivable. Recalling that F is an expectation (1.5), we could apply one of the derivative-free optimization algorithm for general stochastic functions discussed in Section 1.3. However, since these methods would create SAA estimates of F for each iterate and trial point anew without reusing earlier function evaluations, they are expected to perform significantly worse and are not explored here. We could also compare against other regression-based versions of the basic trust-region DFO method. Such a method would compute the model via some kind of regression similar to (2.14) instead of the interpolation in our baseline method, with the expectation that this would smooth out some of the noise. Here, different ways to choose the regression weights are possible; for example, giving equal weights to all points within a certain radius or using the heuristic suggested in [5]. We are not aware of theoretical convergence guarantees for those methods, in contrast to the framework proposed here. A thorough comparison of different methods for handling

computational noise is out of the scope of this paper. The goal of our experiments is to show that our improvements of the baseline algorithm indeed increase the quality of the solutions obtained.

4.1. Algorithm details.

4.1.1. Statistical test. Step 4 of Algorithm 1 requires the computation of a confidence interval $\mathcal{C}_k$ for the difference $\delta_k = F(x_k) - F(s_k)$. Recall that the estimates of F are computed using self-normalized multiple importance sampling:

$$\tilde{F}^{\mathcal{Y}}(x_k) = \sum_{(t_i,y_i)\in\mathcal{Y}} \bar{w}_\sigma(t_i, y_i; x_k) f(y_i), \qquad \tilde{F}^{\mathcal{Y}}(s_k) = \sum_{(t_i,y_i)\in\mathcal{Y}} \bar{w}_\sigma(t_i, y_i; s_k) f(y_i),$$

where the weights $\bar{w}_\sigma(t_i, y_i; x_k)$ are given in (2.8).

Our implementation follows the statistical test proposed in [9]. In contrast to [9], the estimates of the two values $F(x_k)$ and $F(s_k)$ are not independent because the samples $\mathcal{Y}$ are common to both estimates. We argue that the resulting confidence interval is more conservative in our case since the estimates $\tilde{F}^{\mathcal{Y}}(x_k)$ and $\tilde{F}^{\mathcal{Y}}(s_k)$ are built on the same sample points and are therefore positively correlated, at least in the relevant asymptotic case when x_k and s_k are close to each other.

Following [9], we compute the variance of the estimate of δ as $\tilde{\sigma}_\delta^2 = \frac{\tilde{\sigma}_x^2}{n_{e,x}} + \frac{\tilde{\sigma}_s^2}{n_{e,s}}$. The quantity $\tilde{\sigma}_x^2$ is the sample variance of the importance sampling estimator $\tilde{\sigma}_x^2 = \sum_{(t_i,y_i)\in\mathcal{Y}} \bar{w}_\sigma(t_i, y_i; x_k)^2 \left(f(y_i) - \tilde{F}^{\mathcal{Y}}(x_k)\right)^2$ and $n_{e,x}$ is the effective sample size of the importance sampling estimator of $F(x_k)$. In accordance to [42, Ch. 9], the effective sample size n_e in an importance sampling setting is given by $n_{e,x} = \frac{n\bar{w}^2}{\overline{w^2}}$, where

$$\bar{w} = \frac{1}{|\mathcal{Y}|} \sum_{(t_i,y_i)\in\mathcal{Y}} \bar{w}_\sigma(t_i, y_i; x_k) \quad \text{and} \quad \overline{w^2} = \frac{1}{|\mathcal{Y}|} \sum_{(t_i,y_i)\in\mathcal{Y}} \bar{w}_\sigma(t_i, y_i; x_k)^2.$$

The quantities $\tilde{\sigma}_s^2$ and $n_{e,s}$ are defined analogously. We then compute the degrees of freedom as $df = \tilde{\sigma}_\delta^4 \left[\frac{(\tilde{\sigma}_x/n_{e,x})^2}{n_{e,x}-1} + \frac{(\tilde{\sigma}_s/n_{e,s})^2}{n_{e,s}-1}\right]^{-1}$, and the confidence interval from $\mathcal{C}_k = \left[\tilde{\delta}_k - \tilde{\sigma}_\delta \cdot t_{1-\frac{\alpha_k}{2},df}; \tilde{\delta}_k + \tilde{\sigma}_\delta \cdot t_{1-\frac{\alpha_k}{2},df}\right]$, where t denotes the critical values of the student's t-distribution and $\tilde{\delta}_k = \tilde{F}^{\mathcal{Y}}(x_k) - \tilde{F}^{\mathcal{Y}}(s_k)$.

Considering the lower bound, we are able to make a decision once we have

$$\tilde{\delta}_k - \tilde{\sigma}_\delta t_{1-\frac{\alpha_k}{2},df} \geq \eta^L \delta_k^m. \tag{4.1}$$

The expected value of $\tilde{\delta}_k$ is $\mathbb{E}[\tilde{\delta}_k] = \delta_k$, and if we start from scratch, the expected value of $\tilde{\sigma}_\delta$ with N samples in Step 4 in Algorithm 1 given our definition is $\mathbb{E}[\tilde{\sigma}_\delta] = \frac{\sigma_x+\sigma_s}{\sqrt{N}}$ where σ_x and σ_s are the true variances corresponding to $\tilde{\sigma}_x$ and $\tilde{\sigma}_s$, respectively. Substituting the expected values into (4.1) shows that the expected number of samples to reach a conclusion about the lower bound (2.22) is $N \geq \left(\frac{(\sigma_x+\sigma_s)t_{1-\frac{\alpha_k}{2},df}}{\delta_k - \eta^L \delta_k^m}\right)^2$. The same results holds for (2.23) with η^L replaced by η^U. We note that, if many points are needed, this is a good indication that we are close to an optimal solution since the steps are now very small. Also, these samples will then be useful for the subsequent iterations.

4.1.2. Improving the conditioning of the regression problem. As discussed at the end of Section 2.3, the matrix $K_\sigma^{\mathcal{Y}}(x_k)$ in the linear system (2.18) might be singular or ill-conditioned when only few sample points are close to x_k. To keep the analysis simple, we assumed in Algorithm 1 that then additional sampling points are chosen around x_k until the condition number is below the threshold $\kappa_{\max}$.

To save function evaluations, in our implementation we first try to reduce the condition number by using a target distribution with larger standard deviation $\sigma_r > \sigma$ in the regression problem (2.14), where we replace the term "$w_\sigma(t_i, y_i; x_k)$" by "$w_{\sigma_r}(t_i, y_i; x_k)$". This increases the weights of further away points, and the matrix in the linear system (2.18), now $K_{\sigma_r}^{\mathcal{Y}}(x_k)$, may be better conditioned. More precisely, when $\operatorname{cond}\big(K_\sigma^{\mathcal{Y}}(x_k)\big) > \kappa_{\max}$ in Step 2, we set $\sigma_r = \gamma_r \sigma$ with a fixed parameter $\gamma_r > 1$ ($\gamma_r = 1.5$ in our implementation), and recompute the weights $w_{\sigma_r}(t_i, y_i; x_k)$ and the matrix $K_{\sigma_r}^{\mathcal{Y}}(x_k)$. If $\operatorname{cond}\big(K_{\sigma_r}^{\mathcal{Y}}(x_k)\big)$ is still larger than $\kappa_{\max}$, then σ_r is further increased by the factor γ_r. This procedure is repeated until finally $\operatorname{cond}\big(K_{\sigma_r}^{\mathcal{Y}}(x_k)\big) \leq \kappa_{\max}$, or until σ_r becomes larger than some fixed threshold $\sigma_{\max}$ ($\sigma_{\max} = 10^4$ in our implementation). In the latter case, we conclude that it is indeed the positioning of the sample points that causes the ill-conditioning of the regression problem, and we resume the original model improvement procedure in Step 2 of Algorithm 1, now drawing samples around x_k according to the original target distribution $\phi_\sigma(\,\cdot\,; x_k)$.

Note that our method is set up to collect an infinite number of sample points around any limit point of the iterates, so that eventually the condition number of $K_\sigma^{\mathcal{Y}}(x_k)$ is automatically smaller than $\kappa_{\max}$; see Lemma 3.4. Therefore, the enhancement described above is not activated for late iterations, and the convergence guarantees of the algorithm are not affected.

4.1.3. Repeated model improvement. If the model m_k is based only on a few sample points close to x_k, it might not provide a good local approximation of the objective function and the resulting step might not be effective. In this situation, it might not be worth spending a lot of sample evaluations on tightening the confidence interval to decide whether a trial point, generated by a bad model, should be accepted or not. Instead, in our implementation, we preempt Step 4 if a given budget of sample evaluations (20 in our implementation) has been exceeded and return to Step 1 to rebuild the model. Having now more local sample points at hand, the resulting trial point might be better because the model m_k predicts F more accurately. The iterate x_k and the trust-region radius Δ_k remain unchanged. The convergence theory in the previous section still applies, because the model parameters ζ_k and therefore the trial point s_k converge if Step 4 is repeatedly preempted. Then the number of sample points around x_k and s_k increase until the confidence interval $\mathcal{C}_k$ is small enough to decide whether a trial point should be accepted.

4.1.4. Fast acceptance of trial points. From a practical point of view, we would like to accept a trial iterate that leads to a significant reduction in the function estimates $\tilde{F}^{\mathcal{Y}}(x_k)$ and $\tilde{F}^{\mathcal{Y}}(s_k)$, without computing a confidence interval. This reduces sampling in the initial stages of the algorithm when the steps are large and the effective sample size n_e might be lower than 2, and the procedure in Section 4.1.1 would trigger the sampling of additional points. It turns out that we can include such a "fast acceptance test" in our algorithm without compromising the convergence properties and without any additional assumptions on the function f (see [28]). More precisely, for a fixed parameter $\eta_{\text{fast}} > 0$ ($\eta_{\text{fast}} = 1$ for our experiments), we accept a trial point s_k, if $\tilde{F}^{\mathcal{Y}}(s_k) \leq \tilde{F}^{\mathcal{Y}}(x_k) - \eta_{\text{fast}}$ and reject it if $\tilde{F}^{\mathcal{Y}}(s_k) \geq \tilde{F}^{\mathcal{Y}}(x_k) + \eta_{\text{fast}}$.

4.2. Baseline DFO algorithm. In the basic DFO framework in [21, 44], the model m_k in (2.1) interpolates the (noisy) function, i.e.,

$$m_k(y_j) = f(y_j) \text{ for all } y_j \in \mathcal{Y} \tag{4.2}$$

for some set of interpolation points $\mathcal{Y}$. If $|\mathcal{Y}| = (n+1)(n+2)/2$, the parameters of the quadratic model m_k are uniquely defined (assuming poisedness of $\mathcal{Y}$) and can be obtained by solving a linear system of equations. To save expensive function evaluations at the early stages of the optimization, the algorithm starts with a smaller set of interpolation points and resolves the degrees of freedom by computing the model coefficients from

$$(b_k, g_k, H_k) \in \operatorname*{argmin}_{(b,g,H)} \{\|H\|_F^2, \text{ s.t. } (4.2)\}, \tag{4.3}$$

where $\|\cdot\|_F$ is the Frobenius norm, until enough sample points have been accumulated to define a unique interpolation model.

The basic DFO algorithm abstains from performing a model improvement step that ensures well-poisedness of the interpolation points. The role of the geometry of the interpolation points in model-based algorithms for derivative-free optimization has been discussed in [15, 44]. Experiments have demonstrated that simple derivative-free algorithms without geometry verification subroutines, such as the one below, still perform well [21, 44], even though convergence cannot be guaranteed.

The basic DFO algorithm used in our numerical comparison is the following:

ALGORITHM 3 (Basic DFO algorithm).

Parameters: *Choose constants* $\eta_L \in (0,1)$, $0 < \gamma_{\text{dec}} < 1 < \gamma_{\text{inc}}$, *and* $\Delta_{\max} > 0$.

Step 0: Initialization.
Set $k \leftarrow 0$ *and select an initial starting point* x_0 *and trust-region radius* Δ_0. *Let* $\mathcal{Y}_0$ *be a set of* $2n+1$ *initial sample points.*

Step 1: Model construction. *Build an interpolation model* m_k *using* $\mathcal{Y}_k$*: If* $|\mathcal{Y}_k| < (n+1)(n+2)/2$, *solve* (4.3)*; otherwise* $|\mathcal{Y}_k| = (n+1)(n+2)/2$ *and* (4.2) *uniquely determines* m_k.

Step 2: Compute a trial point. *Solve the trust-region subproblem* (2.2), *yielding a solution* p_k, *and let* $s_k := x_k + p_k$.

Step 3: Evaluate tentative point. *Compute the ratio* $\rho_k := \frac{f(x_k) - f(s_k)}{m_k(x_k) - m_k(s_k)}$.

Step 4: Update. *Augment the set of sample points* $\tilde{\mathcal{Y}}_{k+1} = \mathcal{Y}_k \cup \{s_k\}$.

Successful iteration: *If* $\rho_k \geq \eta_L$, *let* $x_{k+1} = s_k$ *and* $\Delta_{k+1} = \min\{\gamma_{\text{inc}}\Delta_k, \Delta_{\max}\}$.

Unsuccessful iteration: *If* $\rho_k < \eta_L$, *let* $x_{k+1} = x_k$, $\Delta_{k+1} = \gamma_{\text{dec}}\Delta_k$.

Remove most-remote interpolation point: *If* $|\tilde{\mathcal{Y}}_{k+1}| > (n+1)(n+2)/2$, *set* $\mathcal{Y}_{k+1} = \tilde{\mathcal{Y}}_{k+1} \setminus \{y_{k,\max}\}$, *where* $y_{k,\max} \in \arg\max_{y \in \tilde{\mathcal{Y}}_{k+1}} \|y - x_{k+1}\|$.

Let $k \leftarrow k+1$ *and return to step 1.*

4.3. Implementation details. Both the proposed method (Algorithm 1) and the basic DFO method (Algorithm 3) were implemented in Matlab and executed (in single-core mode) on a Linux/Ubuntu workstation with eight 3.4GHz Intel Core i7 processors and 32GB RAM. The trust region subproblems (2.2) were solved exactly. The compact set C was not explicitly imposed; i.e., we implicitly chose C large enough to encompass all iterates.

To make the initial phase of the optimization more efficient, our implementation of Algorithm 1 constructs an interpolation model m_k from (4.3) until the set $\mathcal{Y}$ of sample points has at least $(n+2)(n+1)/2$ points, in analogy to Step 1 in Algorithm 3. For both methods, the $2n+1$ initial sample points were sampled according to $\mathcal{N}(x_0, \sigma^2 I)$ and $\mathcal{N}(x_0 \pm \Delta_0 e_i, \sigma^2 I)$ around the starting point x_0. As a consequence, both methods perform identically in the first few iterations due to the fast acceptance test described in Section 4.1.4.

For our numerical experiments, we used the following values for the shared parameters: $\Delta_0 = \max\{1, \|x_0\|_\infty\}$, $\Delta_{\max} = 10^4$, $\gamma_{\text{dec}} = 0.5$, $\gamma_{\text{inc}} = 2$, and $\eta_L = 10^{-8}$. We additionally let $\Delta_{\text{reset}} = 1$, $\eta_U = 0.01$, $\alpha_k = 0.5 \times 0.999^k$, $\kappa_{\max} = 10^{12}$, and $\sigma = 0.1$ for Algorithm 1. In our implementation of Step 4 of Algorithm 1, only a single sample point is added whenever the confidence interval needs to be improved. If $n_{e,x} < n_{e,s}$, the new sample is drawn from $\mathcal{N}(x_k, \sigma^2 I)$, otherwise from $\mathcal{N}(s_k, \sigma^2 I)$.

4.4. Numerical results. In this section, we compare the performance of Algorithm 1 to that of the baseline Algorithm 3 on a set of benchmark problems.

4.4.1. Benchmark problems. Our numerical tests follow closely the benchmark by Moré and Wild [32], which explores the performance of derivative-free algorithms for problems with noise. In particular, we use the same test problems (with their Matlab implementation [31]), each defined by a starting point x_0 and a smooth function $\bar{f}$. The original benchmark set in [32] contains 53 different problems, with up to $n = 12$ variables. Information on the test functions, such as their minima, can be found in [30] and [37]. Moré and Wild suggest the model $f(x) = (1 + \varepsilon_f \theta(x))\, \bar{f}(x)$ to imitate the behavior of deterministic noise, where θ is a rapidly oscillating function with range $[-1, 1]$. We set the relative noise level to $\varepsilon_f = 10^{-1}$.

The test set includes problems that are designed to stress the limits of robust (deterministic) derivative-free optimization software. Some functions are purposely badly scaled or have a vast range caused by exponential terms. Because the primary purpose of the proposed algorithm is to handle deterministic computational noise, we are not interested here in assessing the robustness of the method with respect to difficulties induced by bad scaling. Therefore, before applying the noise, we modify some the given smooth function $\bar{f}(x)$ for our tests to ensure that they are in the range $[1, 10^3]$. Specifically, denoting with $\bar{f}_*$ the optimal solution of a smooth test function, we set $\nu = \frac{\bar{f}(x_0) - \bar{f}_*}{999}$ if $\bar{f}(x_0) - \bar{f}_* > 10^3$, and $\nu = 1$ otherwise. We then define the final noisy function with a modified smooth function $\bar{f}(x) \leftarrow \frac{\bar{f}(x) - \bar{f}_*}{\nu} + 1$.

Finally, some of the test problems have initial values $\bar{f}(x_0)$ that are within noise level of the optimal value and were thus removed. Other objective functions involve rapidly growing exponential terms so that the values at some initial sampling points are larger than 10^9 or led to evaluation errors (returning Inf), and were also removed. The final benchmark set $\mathcal{P}$ consisted of 42 problems.

4.4.2. Performance measure. The performance metric we seek to compare is the level of precision reached by an algorithm with a given budget of function evaluations. For this purpose, Moré and Wild introduced the concept of *data profiles* [32]. Given a set of test problems $\mathcal{P}$, these profiles report the fraction of problems solved to a given tolerance τ as the number of function evaluations increases. Here we say that a point x is a solution of a given tolerance $\tau > 0$ if it satisfies

$$\bar{f}(x_0) - \bar{f}(x) + 2\varepsilon_f \tilde{f}_* \geq (1 - \tau)(\bar{f}(x_0) - \tilde{f}_*), \tag{4.4}$$

where $\tilde{f}_*$ is the best of the final values returned by any run of the algorithms that are being compared. The term $2\varepsilon_f \tilde{f}_*$ is included to slightly relax the condition, so that function values that differ on the order of the deterministic noise are considered equally good.

Because problems in higher dimensions tend to require more samples to reach a given convergence criterion, the number of function evaluations is scaled by $1/(n_p+1)$ to yield the number of "simplex gradients", where n_p is the dimension of a problem $p \in \mathcal{P}$. For each problem $p \in \mathcal{P}$, let x_k^p be the kth iterate generated by a particular algorithm. For a given tolerance τ, we then define

$$d(N_g) = \frac{1}{|\mathcal{P}|} \left| \{p \in \mathcal{P} : x_k^p \text{ satisfies (4.4) where } k = \lfloor N_g(n_p+1) \rfloor \} \right|.$$

In words, given a number N_g of simplex gradients, we report the fraction of problems that satisfy the convergence test (4.4) at the current iterate. This profile might not be monotonic, in contrast to the original data profile in [32].

4.4.3. Results on the benchmark set. We present data profiles for Algorithm 1 and the baseline DFO algorithm (Algorithm 3) with tolerance levels $\tau = 0.1$ and $\tau = 0.01$. Since Algorithm 1 is random, we performed 10 runs for each problem with different random seeds. Similarly, we executed the baseline algorithm 10 times, with different random seeds for the randomly generated starting point described in Section 4.3.

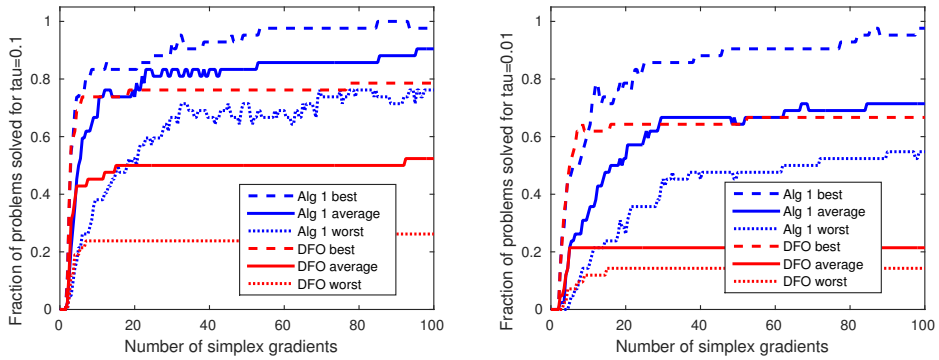

FIG. 4.1. *Data profiles obtained on the benchmark test set for different tolerances τ comparing Algorithm 1 ("Alg 1") and the baseline DFO algorithm ("DFO").*

In Figure 4.1, we display for each algorithm three data profiles that combine the 10 different random runs. They were generated by replacing the value for $\bar{f}(x)$ in (4.4) with the maximum, minimum, and geometric average of the function values for an iteration over the 10 individual runs.

We observe that Algorithm 1 displays a behavior that is very similar to that of the baseline algorithm in the early stage of the optimization. This is due to the fact that the fast acceptance test in Section 4.1.4 enables Algorithm 1 to make progress without the need to establish confidence intervals at the cost of additional function evaluations at the beginning, and the two algorithms consequently take similar steps.

We also see that Algorithm 1 tends to converge to better solutions than the baseline algorithm. It finds the best final objective value within the tolerance $\tau = 0.1$ for almost all problems in at least one of the 10 runs, and the average function values

achieve the same in 90% of the problems. In contrast, the baseline method is able to satisfy this (rather loose) tolerance in only about 75% of the problems, even if one considers the best of all function values encountered over the 10 runs. With this, its best performance is only slightly better than that of the worst function values encountered by Algorithm 1. Going to a tighter tolerance of $\tau = 0.01$, we observe some degradation of the data profiles for Algorithm 1. The baseline algorithm experiences a more dramatic loss. On average, Algorithm 1 still satisfies this tight tolerance in about 70% of the problems, while the baseline method accomplishes this for only 20% of the problems.

This experiment demonstrates that the proposed method tends to avoid getting trapped in local minima caused by noise, and performs similarly to the baseline DFO method when the noise is small compared to large changes in the objective at the early stages of the optimization.

5. Concluding Remarks. We considered the optimization of an objective function that is defined as the convolution of a potentially discontinuous function with a Gaussian kernel. We motivated this formulation as a new approach to handle nonsmooth output of simulation programs caused by deterministic computational noise. The purpose of this formulation is to pose a mathematically well-defined optimization problem and to circumvent the convergence issues that common derivative-free algorithms might encounter when they are applied directly to the noisy function.

We proposed a trust-region algorithm that works with sample average approximations of the convolution. The key ingredient is the use of adaptive multiple importance sampling which permits the utilization of all evaluations of the noisy function from previous iterations. This makes the algorithm a stochastic process for which the convergence analysis is complicated by the fact that the sampling distributions themselves are random variables.

The proposed method is derived as the adaptation of a basic derivative-free optimization algorithm to the minimization of the convolution. It is intended to be competitive with the baseline algorithm in stages where the noise has little impact on the performance of the algorithm and to outperform it in the later stage when the original method breaks down or gets trapped in a spurious local minimum. Numerical experiments, performed on a benchmark set of noisy objective functions, demonstrate that these goals are met.

The choice of the smoothing parameter σ, corresponding to the standard deviation used in the Gaussian kernel, is important. In this paper, we assumed that a good value is known to the practitioner. A sophisticated implementation of our method would include a mechanism to determine a suitable value automatically, potentially based on the method in [33] to estimate computational noise.

We defined the importance sampling weights in (2.7) as a simple likelihood ratios. An alternative is the use of mixture weights [8, 18] given by $w_\sigma^{\mathcal{Y}}(t_i, y_i; x) = \frac{\phi_\sigma(y_i;x)}{\frac{1}{|\mathcal{Y}|}\sum_{(t_j,y_j)\in\mathcal{Y}} \phi_\sigma(y_i;t_j)}$ which are less sensitive to large unbalanced weights caused by small values of $\phi_\sigma(y_i; t_i)$ in (2.7) and have better statistical properties. However, we were not able to prove the uniform convergence results in [49] necessary for the convergence analysis of Algorithm 1 for this alternative definition. We compared the numerical performance of the two definitions in [28] and they behaved similarly for the benchmark set.

Acknowledgments. We thank Katya Scheinberg and Luis Vicente for sharing their Matlab implementation `dfo_tr.m` of Algorithm 3 with us. We are also very grate-

ful to two anonymous referees whose comments helped to improve the presentation of the paper.

REFERENCES

[1] L. K. Alford, R. F. Beck, J. T. Johnson, D. Lyzenga, O. Nwogu, and A. Zundel, *Design, implementation, and evaluation of a system for environmental and ship motion forecasting*, in 30th Symposium on Naval Hydrodynamics, Hobart, Tasmania, Australia, 2-7 November 2014. 3

[2] B. Arouna, *Adaptative Monte Carlo, a variance reduction technique*, Monte Carlo Methods and Applications, 10 (2004), pp. 1–24. 7

[3] A. S. Bandeira, K. Scheinberg, and L. N. Vicente, *Computation of sparse low degree interpolating polynomials and their application to derivative-free optimization*, Mathematical Programming, 134 (2012), pp. 223–257. 21

[4] ———, *Convergence of trust-region methods based on probabilistic models*, SIAM Journal on Optimization, (2015). to appear. 4, 5

[5] S. C. Billups, J. Larson, and P. Graf, *Derivative-free optimization of expensive functions with computational error using weighted regression*, SIAM Journal on Optimization, 23 (2013), pp. 27–53. 5, 10, 21

[6] J. Blanchet, C. Cartis, M. Menickelly, and K. Scheinberg, *Convergence rate analysis of a stochastic trust region method for nonconvex optimization*, arXiv preprint arXiv:1609.07428, (2016). 5

[7] J. Borggaard, D. Pelletier, and K. E. Vugrin, *On sensitivity analysis for problems with numerical noise*, in Proc. 9th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 2002. 3

[8] M. F. Bugallo, V. Elvira, L. Martino, D. Luengo, J. Miguez, and P. M. Djuric, *Adaptive importance sampling: the past, the present, and the future*, IEEE Signal Processing Magazine, 34 (2017), pp. 60–79. 2, 4, 7, 8, 27

[9] K.-H. Chang, L. J. Hong, and H. Wan, *Stochastic trust-region response-surface method (STRONG)-a new response-surface framework for simulation optimization*, INFORMS Journal on Computing, 25 (2013), pp. 230–243. 22

[10] R. Chen, M. Menickelly, and K. Scheinberg, *Stochastic optimization using a trust-region method and random models*, tech. rep., http://www.optimization-online.org/DB_FILE/2015/04/4865.pdf, 2015. 4

[11] W. S. Cleveland and C. Loader, *Smoothing by local regression: Principles and methods*, in Statistical theory and computational aspects of smoothing, Springer, 1996, pp. 10–49. 8

[12] A. R. Conn, I. M. Elfadel, W. Molzen Jr, P. O'Brien, P. N. Strenski, C. Visweswariah, and C. Whan, *Gradient-based optimization of custom circuits using a static-timing formulation*, in Proceedings of the 36th annual ACM/IEEE Design Automation Conference, ACM, 1999, pp. 452–459. 2

[13] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust region methods*, SIAM, 2000. 5, 6, 10

[14] A. R. Conn, K. Scheinberg, and P. L. Toint, *A derivative free optimization algorithm in practice*, in Proceedings of 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO, vol. 48, 1998. 4, 8, 10

[15] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Geometry of interpolation sets in derivative free optimization*, Mathematical programming, 111 (2008), pp. 141–172. 24

[16] ———, *Global convergence of general derivative-free trust-region algorithms to first- and second-order critical points*, SIAM Journal on Optimization, 20 (2009), pp. 387–415. 4

[17] ———, *Introduction to derivative-free optimization*, SIAM, 2009. 4, 5, 10

[18] J. Cornuet, J.-M. Marin, A. Mira, and C. P. Robert, *Adaptive multiple importance sampling*, Scandinavian Journal of Statistics, 39 (2012), pp. 798–812. 7, 27

[19] A. Custódio, K. Scheinberg, and L. Vicente, *Methodologies and software for derivativefree optimization*, in Advances and Trends in Optimization with Engineering Applications, T. Terlaky, M. Anjos, and S. Ahmed, eds., MOS-SIAM Book Series on Optimization, SIAM, Philadelphia, 2017, ch. 37. 4

[20] G. Deng and M. C. Ferris, *Adaptation of the UOBYQA algorithm for noisy functions*, in Proceedings of the 38th conference on Winter simulation, Winter Simulation Conference, 2006, pp. 312–319. 5

[21] G. Fasano, J. L. Morales, and J. Nocedal, *On the geometry phase in model-based algorithms for derivative-free optimization*, Optimization Methods & Software, 24 (2009), pp. 145–154. 21, 24

[22] P. Glasserman, *Monte Carlo methods in financial engineering*, Springer, 2004. 6
[23] R. Hooke and T. A. Jeeves, *"direct search" solution of numerical and statistical problems*, Journal of the ACM, 8 (1961), pp. 212–229. 3
[24] J. Jacod and P. E. Protter, *Probability essentials*, Springer, second ed., 2004. 20
[25] C. T. Kelley, *Iterative Methods for Optimization*, SIAM, 1999. 3
[26] ———, *Implicit Filtering*, SIAM, 2011. 3
[27] J. M. Larson and S. C. Billups, *Stochastic derivative-free optimization using a trust region framework*, tech. rep., http://www.optimization-online.org/DB_FILE/2013/07/3978.pdf, 2013. 5
[28] A. Maggiar, *Optimization of Smoothed Functionals and Applications of Nonlinear Programming to Fastest Path Finding for Vehicles in Anisotropic Media*, PhD thesis, Northwestern University, 2014. 3, 15, 23, 27
[29] M. Marazzi and J. Nocedal, *Wedge trust region methods for derivative free optimization*, Mathematical programming, 91 (2002), pp. 289–305. 8
[30] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, *Testing unconstrained optimization software*, ACM Transactions on Mathematical Software (TOMS), 7 (1981), pp. 17–41. 25
[31] J. J. Moré and S. M. Wild, *Benchmarking derivative-free optimization algorithms*. http://www.mcs.anl.gov/ more/dfo/. 25
[32] ———, *Benchmarking derivative-free optimization algorithms*, SIAM Journal on Optimization, 20 (2009), pp. 172–191. 4, 21, 25, 26
[33] ———, *Estimating computational noise*, SIAM Journal on Scientific Computing, 33 (2011), pp. 1292–1314. 3, 4, 27
[34] ———, *Estimating derivatives of noisy simulations*, ACM Transactions on Mathematical Software (TOMS), 38 (2012), p. 19. 3
[35] A. Nedić and D. P. Bertsekas, *The effect of deterministic noise in subgradient methods*, Mathematical programming, 125 (2010), pp. 75–99. 3
[36] J. A. Nelder and R. Mead, *A simplex method for function minimization*, The Computer Journal, 7 (1965), pp. 308–313. 3
[37] H. B. Nielsen, *Uctp-test problems for unconstrained optimization*, tech. rep., Informatics and Mathematical Modelling (IMM), Technical University of Denmark, 2000. 25
[38] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, Springer, second ed., 2006. 5
[39] O. G. Nwogu, *Interaction of finite-amplitude waves with vertically-sheared current fields*, J. of Fluid Mechanics, 627 (2009), pp. 179–213. 3
[40] O. G. Nwogu and D. R. Lyzenga, *Surface wavefield estimation from coherent marine radars*, IEEE Geoscience and Remote Sensing Letters, 7 (2010), pp. 631–635. 3
[41] A. Owen and Y. Zhou, *Safe and effective importance sampling*, Journal of the American Statistical Association, 95 (2000), pp. 135–143. 2, 4, 8
[42] A. B. Owen, *Monte Carlo theory, methods and examples*, http://statweb.stanford.edu/~owen/mc/, 2013. 8, 22
[43] M. J. D. Powell, *UOBYQA: unconstrained optimization by quadratic approximation*, Mathematical Programming, 92 (2002), pp. 555–582. 5, 8
[44] K. Scheinberg and P. L. Toint, *Self-correcting geometry in model-based algorithms for derivative-free unconstrained optimization*, SIAM Journal on Optimization, 20 (2010), pp. 3512–3532. 21, 24
[45] S. Shashaani, F. Hashemi, and R. Pasupathy, *Astro-df: A class of adaptive sampling trust-region algorithms for derivative-free stochastic optimization*, arXiv preprint arXiv:1610.06506, (2016). 5
[46] E. Veach and L. J. Guibas, *Optimally combining sampling techniques for Monte Carlo rendering*, in Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, ACM, 1995, pp. 419–428. 2, 4
[47] C. Visweswariah and R. A. Rohrer, *Piecewise approximate circuit simulation*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 10 (1991), pp. 861–870. 2
[48] K. E. Vugrin, *On the effects of noise on parameter identification optimization problems*, PhD thesis, Virginia Polytechnic Institute and State University, 2005. 3
[49] A. Wächter, J. Staum, A. Maggiar, and M. Feng, *Sample average approximation with adaptive importance sampling*, tech. rep., IEMS Department, Northwestern University, October 2017. 14, 15, 27
[50] X. Zhang, P. Bandyk, and R. F. Beck, *Seakeeping computations using double-body basis flows*, Applied Ocean Research, 32 (2010), pp. 471–482. 2, 3