Stochastic Black-Box Optimization using Multi-Fidelity Score Function Estimator

Atul Agrawal, Phaedon-Stelios Koutsourelakis, Kislaya Ravi, Hans-Joachim Bungartz

Stochastic Black-Box Optimization using Multi-Fidelity Score Function Estimator

Atul Agrawal*, **Phaedon-Stelios Koutsourelakis**
Professorship of Data-Driven Materials Modelling
Technical University of Munich
Munich, Germany
{atul.agrawal,p.s.koutsourelakis}@tum.de

Kislaya Ravi*, **Hans-Joachim Bungartz**
Chair of Scientific Computing
Technical University of Munich, Germany
{kislaya.ravi,bungartz}@tum.de

Abstract

Optimizing the parameters of physics-based simulators is crucial in the design process of engineering and scientific systems. This becomes challenging when the simulator is stochastic, computationally expensive, black-box, multi-modal, and has a high-dimensional parameter space, such as when simulating complex climate models that involve numerous interacting variables and uncertain parameters. Many traditional optimization methods rely on gradient information, which is frequently unavailable in legacy black-box codes. To address these challenges, we present SCOUT-Nd (Stochastic Constrained Optimization for N dimensions), an algorithm designed to efficiently estimate gradients, reduce noise in the gradient estimator, and enhance convergence properties through the use of natural gradients. SCOUT-Nd also incorporates multi-fidelity schemes and an adaptive number of samples to minimize computational effort. We validate our approach using standard benchmark analytical problems, demonstrating its superior performance in parameter optimization compared to existing methods. Additionally, we showcase the algorithm's efficacy in a complex real-world application: optimizing wind farm layout.

1 Introduction

Several real-world continuous optimization problems are too difficult to solve due to the involvement of complex physics-based simulators in the objective or the constraints. These physics-based simulators are used across fields of engineering and science to drive research [1]. There are many problems that fit into this category, spanning a wide array of fields such as aeronautic design [2, 3], applications in healthcare and science [4, 5], material design [6, 7], optimizing particle-physics instruments [8], optimizing wind-farm layouts [9], general physics [10, 11], reinforcement learning (RL) [12] and generating synthetic training data for machine learning related tasks [13, 14].

We consider a high-dimensional and expensive to a query function $f : \mathbb{R}^d \to \mathbb{R}$. The high-dimensional parameter space, multiple local optima, the lack of gradients, and stochasticity in the objective function evaluation make the optimization non-trivial. This setting is prevalent in science and engineering domains [15]. Optimization is primarily facilitated through gradient-based or

*equal contribution

Preprint. Under review.

gradient-free ("black-box") methods [16]. Gradient-based methods are effective when derivatives are readily available [17, 5, 18, 19]. Over the past decade, the surge in interest in machine learning (ML) has significantly propelled the field of differentiable programming [16]. However, in most real-world optimization scenarios involving physics-based simulators, gradients are not available inherently [20]. They are typically obtained through one of several strategies: a) employing the adjoint method [21, 5, 22], b) rewriting the simulator in a differentiable programming language such as JAX, PyTorch, Julia, etc. [23, 24], or c) developing a differentiable surrogate, e.g. based on neural networks, to approximate the objective function (e.g., [8]). The first two options pose significant implementation challenges, often involving extensive modifications to legacy codes, thus impeding applying ML to scientific computing [25]. Furthermore, training a differentiable surrogate can be prohibitively expensive due to the initial cost of data generation. Also, the effectiveness of the optimization is contingent upon the quality of the surrogate model developed. Recently, automatic differentiable (AD) compiler methods [26] have emerged, providing gradients of legacy codes written in C, C++, Fortran, etc. These methods facilitate integration with existing machine learning infrastructure and support gradient-based optimization. However, these approaches are not directly comparable to the methods proposed in the current study.

The gradient-free methods [27, 28, 29] are used for optimization when only black-box evaluations are possible. It is also commonly called simulation-based inference(SBI)/optimization [1, 30]. Some widely used methods include genetic algorithms [31], Bayesian Optimization and their extensions [32, 33], and Evolution strategy (ES) methods [34, 35, 36], to name a few. The gradient-free methods perform poorly on high-dimensional parametric spaces [29]. To remedy this, recently, stochastic gradient estimators [37] have been used to estimate gradients of black-box functions and, hence, perform gradient-based optimization [38, 39, 10, 13, 40]. However, they do not account for the constraints. Also, since the methods rely on Monte Carlo techniques, the computational cost can be very high due to many expensive simulator calls, and the gradients can be very noisy.

This work introduces a novel gradient-free approach SCOUT-Nd (Stochastic Constrained Optimization for N Dimensions) and MF-SCOUT-Nd (Multi-Fidelility Stochastic Constrained Optimization for N Dimensions) for constrained stochastic optimization involving stochastic black-box simulators with high-dimensional parametric dependency. As per the notion of oracles [41, 16], the proposed algorithm uses a stochastic, zeroth-order oracle [42]. We draw inspiration from Variational Optimization [43, 44] to estimate the gradients, provide extensions that account for constraints, and employ multi-fidelity strategies to improve efficiency by incorporating lower-fidelity and less expensive simulator(s). The proposed algorithm consists of the following major elements: (a) A non-intrusive method to estimate gradients of black-box physical simulators (Sec. 2.2), with an ability to account for stochasticity in the objective (Sec. 2) and handle constraints using penalty methods (Sec. 2.1). (b) Strategies to reduce the variance of the gradient estimator (Sec. 2.3). (c) Ability to handle non-convexity (Sec. 2.6). (d) Better optimum and well-behaved convergence properties using natural gradients (Sec. 2.4). (e) Multi-fidelity strategies (Sec. 2.7) and adaptive selection of the number of samples for gradient estimation (Sec. 2.6.4 and Sec. 2.7.1) to provide trade-off between computational cost and accuracy. The structure of the rest of the paper is as follows. Sec. 2 defines the problem we address in the present work and the proposed algorithm with relevant details and analysis. In Sec. 3, we present the state-of-the-art performance of SCOUT-Nd/MF-SCOUT-Nd on standard benchmark analytical problems and compare the results with popular gradient-free (constrained) optimization methods like Constrained Bayesian Optimization (cBO)[45], COBYLA [46] and SLSQP[47]. Secondly, we test our algorithms on a real-world, expensive, physics-based simulator. We choose the windfarm layout optimization case [15], which, owing to the absence of derivatives, presence of constraints, multi-model surface, and stochasticity, provides a challenging test case for black-box optimization routines [48]. We compare our results with SLSQP. In Sec. 4, we summarize our findings and discuss limitations and potential enhancements.

2 Methodology

Problem statement We are concerned with optimizing a scalar-valued function $f(\boldsymbol{x}, \boldsymbol{b})$ subject to constraints $\boldsymbol{\mathcal{C}}(\boldsymbol{x}) = \{\mathcal{C}_1(\boldsymbol{x}), \ldots, \mathcal{C}_I(\boldsymbol{x})\}$, where $\boldsymbol{x} \in \mathbb{R}^d$ denote the potentially high-dimensional deterministic parameters and $\boldsymbol{b}$ represents uncertain parameters [38]. A lack of knowledge about the parameters or the inherent noise in the system may cause uncertainty. The objective f or the constraints $\mathcal{C}$ depend implicitly on the output of the black-box simulator, thus we only have access to

a zero-order oracle. Since the solution of the optimization problem to obtain the optimal design $\boldsymbol{x}$ involves the random vector $\boldsymbol{b}$, its variability needs to be involved in the optimization process to limit its negative effect on the optimal design. This is classically done using a robustness measure [49, 50] given by $\mathcal{R}$. The general parameter-dependent nonlinear constrained optimization problem can be stated as

$$\min_{\boldsymbol{x}} \mathcal{R}[f(\boldsymbol{x}, \boldsymbol{b})], \quad \text{s.t} \;\; \mathcal{C}_i(\boldsymbol{x}) \leq 0, \;\; \forall i \in \{1, \ldots, I\}. \tag{1}$$

In this work, we will only consider the expectation as a robustness measure, in which case, the problem can be stated as follows:

$$\min_{\boldsymbol{x}} \mathbb{E}_{\boldsymbol{b}}[f(\boldsymbol{x}, \boldsymbol{b})], \quad \text{s.t} \;\; \mathcal{C}_i(\boldsymbol{x}) \leq 0, \;\; \forall i \in \{1, \ldots, I\}. \tag{2}$$

In addition to that, the gradient of the objective function and the constraint is unavailable. Hence, one cannot directly apply gradient-based optimization methods.

2.1 Penalizing the constraints

To tackle the constrained optimization problem (Eq. (2)), we convert it to an unconstrained one using penalty-based methods [51, 52]. We define an augmented objective function $\mathcal{L}$ as follows:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda}) = f(\boldsymbol{x}, \boldsymbol{b}) + \sum_{i=i}^{I} \lambda_i \max\left(\mathcal{C}_i(\boldsymbol{x}), 0\right), \tag{3}$$

where $\lambda_i > 0$ is the penalty parameter for the i^{th} constraint and the $\max(\cdot, \cdot)$ controls the magnitude of the penalty applied. One can make the enforcement of a particular constraint stricter by increasing the value of the corresponding penalty parameter. Incorporating the augmented objective (Eq. (3)) in Eq. (2), one can arrive at the following optimization problem:

$$\min_{\boldsymbol{x}} \mathbb{E}_{\boldsymbol{b}}[\mathcal{L}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda})]. \tag{4}$$

Optimization The Monte Carlo method approximates the expectation, which induces noise. We alleviate the dependence on the penalty parameter $\boldsymbol{\lambda}$ by using the sequential unconstrained minimization technique (SUMT) algorithm [53]. The algorithm considers a strictly increasing sequence $\{\boldsymbol{\lambda}_n\}$ with $\boldsymbol{\lambda}_n \to \infty$. [53] show that when $\boldsymbol{\lambda}_n \to \infty$, then the sequence of corresponding minima, say $\{\boldsymbol{x}_n^*\}$, converges to a global minimizer $\boldsymbol{x}^*$ of the original constrained problem. The SUMT technique has also been shown to work with non-linear constraints [54]. This adaptation of the penalty parameters helps to balance the need to satisfy the constraints with the need to make progress towards the optimal solution. Augmented Lagrangian [55] represents an alternative approach that could be considered for application in this context, though it has not been explored in the current study.

2.2 Gradient Estimation

Since the design variable $\boldsymbol{x}$ can be high-dimensional, gradient-free methods such as genetic programming, Bayesian Optimization, etc. may not be as efficient as gradient-based, and the latter should be used whenever available [28]. Unfortunately, the direct computation of derivatives of $\mathcal{L}$ with respect to the optimization variables $\boldsymbol{x}$ is not feasible because of the unavailability of the gradients of the objective function and the constraints. One notes many active research threads across disciplines are trying to tackle this bottleneck of unavailability of gradients in physical simulators [1, 39, 56, 57, 5, 10]. We draw inspiration from the Variational Optimization [43, 58, 13], which constructs an upper bound of the objective function as shown below:

$$\min_{\boldsymbol{x}} \int \mathcal{L}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda}) p(\boldsymbol{b}) \, \mathrm{d}\boldsymbol{b} \leq \int \mathcal{L}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda}) p(\boldsymbol{b}) q(\boldsymbol{x} \mid \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{b} \, \mathrm{d}\boldsymbol{x} = U(\boldsymbol{\theta}), \tag{5}$$

where $q(\boldsymbol{x} \mid \boldsymbol{\theta})$ is a probability density over the design variables $\boldsymbol{x}$ with parameters $\boldsymbol{\theta}$. If $\boldsymbol{x}^*$ yields the minimum of the objective $\mathbb{E}_{\boldsymbol{b}}[\mathcal{L}]$, then this can be achieved with a degenerate q that collapses to a Dirac-delta, i.e. if $q(\boldsymbol{x} \mid \boldsymbol{\theta}) \approx \delta(\boldsymbol{x} - \boldsymbol{x}^*)$. The inequality above would generally be strict for all other densities q or parameters $\boldsymbol{\theta}$. Hence, instead of minimizing $\mathbb{E}_{\boldsymbol{b}}[\mathcal{L}]$ with respect to $\boldsymbol{x}$, we can minimize the upper bound $U(\boldsymbol{\theta})$ with respect to the distribution parameters $\boldsymbol{\theta}$. Under mild restrictions

outlined by [44], the bound $U(\boldsymbol{\theta})$ is differential w.r.t $\boldsymbol{\theta}$. One can evaluate the gradient of $U(\boldsymbol{\theta})$ using the 'log-derivative trick' [37] as shown below:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x},\boldsymbol{b}}[\mathcal{L}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda})] \\
&= \nabla_{\boldsymbol{\theta}} \int q(\boldsymbol{x} \mid \boldsymbol{\theta}) p(\boldsymbol{b}) \mathcal{L}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda}) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{b} \\
&= \int \nabla_{\boldsymbol{\theta}} q(\boldsymbol{x} \mid \boldsymbol{\theta}) p(\boldsymbol{b}) \mathcal{L}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda}) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{b} \\
&= \int q(\boldsymbol{x} \mid \boldsymbol{\theta}) \nabla_{\theta} \log q(\boldsymbol{x} \mid \boldsymbol{\theta}) p(\boldsymbol{b}) \mathcal{L}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda}) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{b} \\
&= \mathbb{E}_{\boldsymbol{x},\boldsymbol{b}} \left[\nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{x} \mid \boldsymbol{\theta}) \mathcal{L}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda}) \right]. \qquad (6)
\end{aligned}$$

As per [44], the expected objective $U(\boldsymbol{\theta})$ conserves the convexity of the original objective, thus helping in the convergence properties. The Monte Carlo estimation of the expectation shown in Eq. (6) is as follows:

$$\frac{\partial U}{\partial \boldsymbol{\theta}} \approx \frac{1}{S} \sum_{i=1}^{S} \mathcal{L}(\boldsymbol{x}_i, \boldsymbol{b}_i, \boldsymbol{\lambda}) \frac{\partial}{\partial \boldsymbol{\theta}} \log q \left(\boldsymbol{x}_i \mid \boldsymbol{\theta} \right). \qquad (7)$$

Eq. (6) is known as the score function estimator [59]. In a manner similar to the REINFORCE [12], we take gradient steps on $\boldsymbol{\theta}$. The score function estimator also appears in the context of reinforcement learning [60]. In the present work, we work with functions with continuous domains, so we use Gaussian for $q(\cdot)$. For the special case where $q(\boldsymbol{x} \mid \boldsymbol{\theta})$ is factored Gaussian, the resulting gradient estimator is also known as *parameter-exploring policy gradients* [61], or *zero-order gradient estimation* [62]. The number of samples S per iteration is usually of the order $\mathcal{O}(d)$ [35, 60] (also demonstrated in Sec. 2.6.4). This can be problematic for high-dimensional problems. Fortunately, the gradient estimation can be embarrassingly parallelized. One needs to synchronize random seeds between machines before optimization, i.e., each machine knows what perturbations the other machine used, so each machine only needs to communicate a single scalar to and from the other machine to agree on a parameter update. This approach also ensures that gradient estimation is non-intrusive, meaning that the physics-based simulator does not need modification to accommodate the optimization routine. Consequently, legacy simulators can be used without any adjustments.

2.3 Variance reduction

Monte Carlo gradient estimation suffers from high variance. Extensive work has been done in this regard in the past decades [37]. In the present work, we propose using the baseline method discussed in [63] to reduce the mean square error of the estimator in Eq. (7). This is given by:

$$\frac{\partial U}{\partial \boldsymbol{\theta}} \approx \frac{1}{S} \sum_{i=1}^{S} \frac{\partial}{\partial \boldsymbol{\theta}} \log q \left(\boldsymbol{x}_i \mid \boldsymbol{\theta} \right) \left(\mathcal{L}(\boldsymbol{x}_i, \boldsymbol{b}_i, \boldsymbol{\lambda}) - \frac{1}{S-1} \sum_{j=1, j \neq i}^{S} \mathcal{L}(\boldsymbol{x}_j, \boldsymbol{b}_j, \boldsymbol{\lambda}) \right). \qquad (8)$$

The above is an unbiased estimator with no additional cost beyond the S samples.

We also propose to use Quasi-Monte Carlo (QMC) sampling [64] for variance reduction, thus promising a more accurate gradient estimate [65]. QMC replaces S randomly drawn samples with a pseudo-random sequence of samples of length S with low discrepancy. This sequence covers the underlying design space more evenly than the random samples, thereby reducing the variance of the gradient estimator. Also, it has been shown that under certain conditions, QMC ($\mathcal{O}(S^{-1})$) reaches a faster rate of convergence as compared to random sampling ($\mathcal{O}(S^{-1/2})$). From a theoretical point of view, the benefit of QMC vanishes in very high dimensions. However, [66] showed that the gains are observed up to dimension 150 in practice. In this work, we present the results using Sobol points [67].

2.4 Natural Gradients

We observe parametric oscillations in the optimization of the 2D Ackley function around the optima, as can be seen in Fig. 1(a) and Fig. 1(b). The Ackley function is widely used to study optimization

algorithms due to its complex, multi-modal landscape. These oscillations occur due to moving in the parametric space $\boldsymbol{\theta}$ using the gradient descent with Euclidean distance as the distance measure. This makes the update dependent on the particular parameterization of the distribution $q(\boldsymbol{x} \mid \boldsymbol{\theta})$. Therefore, a change in parameterization leads to different gradients and different updates. Hence, the optimization is more difficult, and step size reduction with increasing iterations is crucial for stability [68, 69]. To resolve this, we utilize *natural gradients* [68, 34], which relies on a more 'natural' measure of distance $D_{\mathrm{KL}}(\boldsymbol{\theta}' \| \boldsymbol{\theta})$ between the distributions $q(\boldsymbol{x}|\boldsymbol{\theta})$ and $q(\boldsymbol{x}|\boldsymbol{\theta}')$, with $D_{\mathrm{KL}}(\boldsymbol{\theta}' \| \boldsymbol{\theta})$ denoting the Kullback-Leibler divergence [70]. The natural gradient is then the solution to a constrained optimization problem which involves (for details, please see [34])

$$\boldsymbol{F} = \int q(\boldsymbol{x} \mid \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{x} \mid \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{x} \mid \boldsymbol{\theta})^{\top} \, \mathrm{d}\boldsymbol{x} \tag{9}$$

$$= \mathbb{E}_{\boldsymbol{x}} \left[\nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{x} \mid \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{x} \mid \boldsymbol{\theta})^{\top} \right], \tag{10}$$

where $\boldsymbol{F}$ is the *Fisher Information Matrix* (FIM) of the given parametric family of search distributions. As mentioned earlier, in SCOUT-Nd/MF-SCOUT-Nd, the q takes the form of a Multivariate Normal (MVN) with $\boldsymbol{\theta} := (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\boldsymbol{\Sigma} := \mathrm{diag}(\sigma_1^2, \ldots, \sigma_d^2)$. Using the analytical derivative of the MVN, we obtain the following expression for the FIM:

$$\boldsymbol{F}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathrm{diag}\left(\frac{1}{\sigma_1^2}, \cdots, \frac{1}{\sigma_d^2}, \frac{2}{\sigma_1^2}, \cdots, \frac{2}{\sigma_d^2} \right). \tag{11}$$

We use $\sigma_i^2 = e^{2\beta_i}$, reparametrized FIM is given by:

$$\boldsymbol{F}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathrm{diag}(e^{-2\beta_1}, \cdots, e^{-2\beta_d}, 2, \cdots, 2). \tag{12}$$

If the $\boldsymbol{F}$ matrix is invertible, the Monte Carlo gradient estimate is updated to

$$\tilde{\nabla}_{\boldsymbol{\theta}} U = \boldsymbol{F}^{-1} \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}). \tag{13}$$

The gradient in the equation above is called the *natural gradient* [68, 71]. As can be seen in the equation above, the $\boldsymbol{F}$ matrix scales the gradients. Therefore, it can also be seen as a step-size adaptation.

The tendency of the natural gradient is to reduce the value of the gradient if the variance is smaller than one. This slows down the convergence speed in the beginning, even after dampening. One needs natural gradients only near the optimum. So, we apply the natural gradients only when the optimizer is near the optimum. We can identify if the optimizer is in the proximity of the optimum when the value of the variance has decreased to a small value (i.e., $\|\boldsymbol{\Sigma}\| \leq \sigma_{\text{nat-grad}}$). Another way suggested by [69] is to apply a dampening constant to the Fisher information matrix, given by

$$\tilde{\boldsymbol{F}} = \boldsymbol{F} + \eta \boldsymbol{I}. \tag{14}$$

The isotropic damping η can be set constant or using an exponential decay approach by

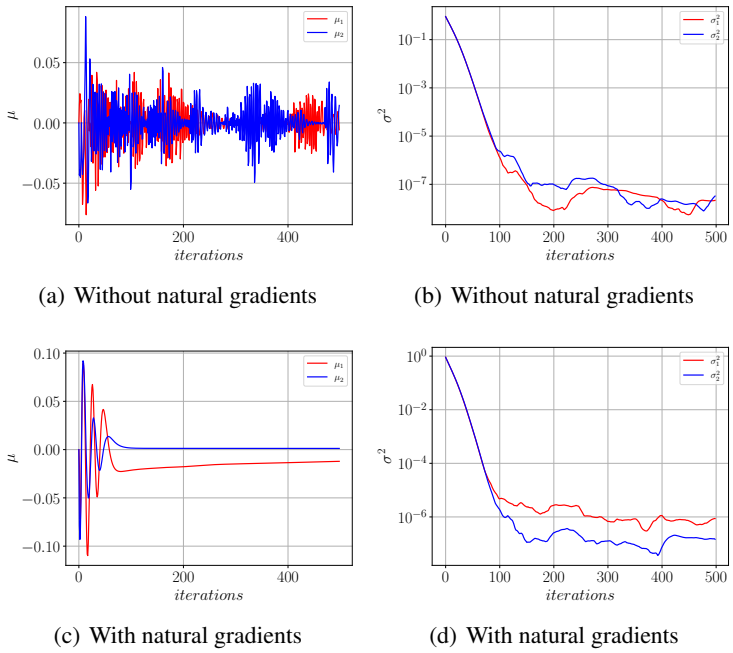
$$\eta = \begin{cases} \tilde{\eta} & \text{if iteration } k < N_{\text{cut-off}}, \\ \max\left(\tilde{\eta} \cdot \exp\left(-\frac{k - N_{\text{cut-off}}}{N_{\text{cut-off}}} \right), \eta_{\text{lower bound}} \right) & \text{else.} \end{cases} \tag{15}$$

In subsequent numerical experiments, the parametric values choose are $\tilde{\eta} = 10^{-1}$, $\eta_{\text{lower bound}} = 10^{-6}$ and $N_{\text{cut-off}} = 100$.

Using the Ackley function, we study the benefits of adding natural gradients to the proposed algorithm. As can be seen in Fig. 1(c) and Fig. 1(d), the evolution of the parameters $\boldsymbol{\theta}$ around the optimum are without oscillations, as opposed to the case when natural gradients are not used (Fig. 1(a) and Fig. 1(b)). This well-behaved parametric evolution also translates to a better optimum, even in high dimensional cases, as seen in Fig. 2.

2.5 Termination criterion

Our proposed algorithms use gradient descent methods to optimize the function. Gradient-based optimization methods converge to a local optimum. Subsequent optimization steps yield only marginal improvements. In such scenarios, continuing the optimization process may be inefficient. It can be beneficial to terminate the algorithm to conserve computational resources. This section explores



(a) Without natural gradients (b) Without natural gradients

(c) With natural gradients (d) With natural gradients

Figure 1: Illustrations highlighting the effect of natural gradients on the 2D Ackley function optimization near the optima.

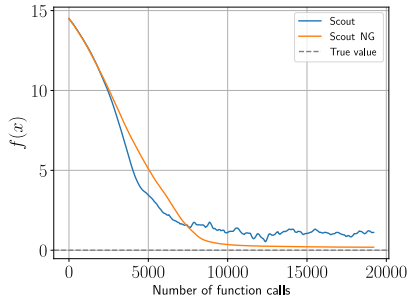


Figure 2: Evolution of Ackley function value with $d = 64$ to highlight the influence of natural gradients. Theoretical $f(\boldsymbol{x}^*) = 0$, learning rate = 0.1, number of samples = 64.

several criteria for terminating the algorithm, addressing both constrained and unconstrained cases separately.

In unconstrained optimization, no penalty term is involved, simplifying the optimization algorithm to a single loop that repeatedly estimates gradients and takes gradient descent steps. We terminate when the computational budget exceeds a predefined limit or the distribution q collapses to a Dirac-delta. For a Gaussian distribution, this occurs when the norm of the variance falls below a specified cut-off value($\sigma_{\text{cut-off}}$). The method proposed for unconstrained optimization problems is summarized in the Algorithm 1.

There are two loops in the algorithm for constrained optimization. The outer loop increases the value of the penalty constant ($\boldsymbol{\lambda}$). The inner loop performs the gradient descent step for a fixed penalty constant. We terminate the outer loop under the same conditions as the unconstrained optimization i.e., when the computational budget runs out or the distribution q collapses to a Dirac-delta. For the inner loop, termination occurs when the computational budget exceeds the predefined limit for a penalty constant or the distribution q collapses to a Dirac-delta. Additionally, we might encounter cases where, for a fixed value of $\boldsymbol{\lambda}$, the optimizer gets stuck at a local optimum in a region where the constraint is not satisfied because the penalty constant is insufficient. This situation can be detected

Algorithm 1: SCOUT-Nd algorithm for unconstrained optimization

Input : Objective function $f(\boldsymbol{x}, \boldsymbol{b})$, distribution $q(\boldsymbol{x} \mid \boldsymbol{\theta})$, gradient descent optimizer $\mathcal{G}$, Natural gradient starting variance $\sigma_{\text{nat-grad}}$, Maximum budget N_{max}, variance cut-off value $\sigma_{\text{cut-off}}$

Set initial point $\boldsymbol{\theta}_0 := \{\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0\}$
Initialize $k \leftarrow 0$
do
  $\boldsymbol{x}_i \sim q(\boldsymbol{x} \mid \boldsymbol{\theta}_k), \quad \boldsymbol{b}_i \sim p(\boldsymbol{b})$ // Sampling step
  Evaluate objective function $f(\boldsymbol{x}_i, \boldsymbol{b}_i)$
  Monte Carlo gradient estimate $\nabla_{\boldsymbol{\theta}} U$ // Eq. (8), 7
  if $\|\boldsymbol{\Sigma}_k\| < \sigma_{\textit{nat-grad}}$ **then**
    Compute the natural gradients $\tilde{\nabla}_{\boldsymbol{\theta}} U$ // Eq. (13)
    $\nabla_{\boldsymbol{\theta}} U \leftarrow \tilde{\nabla}_{\boldsymbol{\theta}} U$
  end
  $\boldsymbol{\theta}_{k+1} \leftarrow \mathcal{G}(\boldsymbol{\theta}_k, \nabla_{\boldsymbol{\theta}} U)$ // Gradient Descent
  $k \leftarrow k + 1$
while $k < N_{\textit{max}}$ *and* $\|\boldsymbol{\Sigma}_k\| > \sigma_{\textit{cut-off}}$
Output : $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$

Algorithm 2: SCOUT-Nd algorithm for constrained optimization

Input : Objective function $f(\boldsymbol{x}, \boldsymbol{b})$, constraints $\boldsymbol{\mathcal{C}}(\boldsymbol{x})$, distribution $q(\boldsymbol{x} \mid \boldsymbol{\theta})$, gradient descent optimizer $\mathcal{G}$, list of penalty terms $\{\boldsymbol{\lambda}_m\}_{m=1}^{M}$, $\boldsymbol{\lambda}_M \to \infty$, Natural gradient starting variance $\sigma_{\text{nat-grad}}$, Maximum budget for inner and outer loop $N_{\text{max}}^{\text{inner}}$, $N_{\text{max}}^{\text{outer}}$, variance cut-off value $\sigma_{\text{cut-off}}$

Set initial point $\boldsymbol{\theta}_0 := \{\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0\}$
Initialize $k \leftarrow 0$, $m \leftarrow 1$
do
  $n \leftarrow 0$
  do
    $\boldsymbol{x}_i \sim q(\boldsymbol{x} \mid \boldsymbol{\theta}_k), \quad \boldsymbol{b}_i \sim p(\boldsymbol{b})$ // Sampling step
    Evaluate augmented objectives $\mathcal{L}(\boldsymbol{x}_i, \boldsymbol{b}_i, \boldsymbol{\lambda}_m)$ // Eq. (3)
    Monte Carlo gradient estimate $\nabla_{\boldsymbol{\theta}} U$ // Eq. (8), 7
    if $\|\boldsymbol{\Sigma}_k\| < \sigma_{\textit{nat-grad}}$ **then**
      Compute the natural gradients $\tilde{\nabla}_{\boldsymbol{\theta}} U$ // Eq. (13)
      $\nabla_{\boldsymbol{\theta}} U \leftarrow \tilde{\nabla}_{\boldsymbol{\theta}} U$
    end
    $\boldsymbol{\theta}_{k+1} \leftarrow \mathcal{G}(\boldsymbol{\theta}_k, \nabla_{\boldsymbol{\theta}} U)$ // Gradient Descent
    $n \leftarrow n + 1$, $k \leftarrow k + 1$
  while $n < N_{\textit{max}}^{\textit{inner}}$ *and* $\|\boldsymbol{\Sigma}_k\| > \sigma_{\textit{cut-off}}$
  if $\exists i : \mathcal{C}_i(\boldsymbol{\mu}_k) > 0$ *and* $\|\boldsymbol{\Sigma}_k\| > \sigma_{\textit{cut-off}}$ **then**
    $\boldsymbol{\Sigma}_k \leftarrow \boldsymbol{\Sigma}_0$
  end
  $m \leftarrow m + 1$
while $k < N_{\textit{max}}^{\textit{outer}}$ *and* $\|\boldsymbol{\Sigma}_k\| > \sigma_{\textit{cut-off}}$
Output : $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$

if the norm of the variance falls below a specified cut-off value ($\sigma_{\text{cut-off}}$) and the constraint is not satisfied. In such cases, we reset the value of the variance term ($\boldsymbol{\Sigma}_k$) to its initial value. The method proposed for constrained optimization problems is summarized in the Algorithm 2.

2.6 Method Analysis

2.6.1 Covergence studies

Our proposed algorithm optimizes $U(\boldsymbol{\theta})$ instead of $f(\boldsymbol{x})$. Let us convert the distribution to standard normal distribution as follows:

$$U(\boldsymbol{\theta}) = \mathbb{E}[f(\boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{z})]_{\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})}, \tag{16}$$

where $\boldsymbol{\sigma}$ is the vector containing the diagonal entries of $\boldsymbol{\Sigma}$. We are ignoring the stochastic term $\boldsymbol{b}$ from the analysis because we assume that the noise term gets marginalized in Eq. (16). Using the Taylor expansion and the simplifications detailed in the Appendix B, we get

$$U(\boldsymbol{\theta}) = f(\boldsymbol{\mu}) + \frac{1}{2} \sum_{i=1}^{d} {\sigma_i}^2 \frac{\partial^2 f}{\partial {x_i}^2}(\boldsymbol{\mu}) + O(\|\boldsymbol{\sigma}\|^4). \tag{17}$$

Taking the derivative of Eq. (17) with respect to $\boldsymbol{\mu}$:

$$\frac{\partial U}{\partial \boldsymbol{\mu}} = \frac{\partial f}{\partial \boldsymbol{\mu}}(\boldsymbol{\mu}) + \frac{1}{2} \sum_{i=1}^{d} {\sigma_i}^2 \frac{\partial^3 f}{\partial {x_i}^2 \partial \boldsymbol{\mu}}(\boldsymbol{\mu}) + O(\|\boldsymbol{\sigma}\|^4). \tag{18}$$

The mean $\boldsymbol{\mu}$ corresponds to the location $\boldsymbol{x}$ in the parameter space. We can ignore the higher order variance term in Eq. (18) for $\|\boldsymbol{\sigma}\| < 1$. So, the difference between $\partial U/\partial \boldsymbol{\mu}$ and $\partial f/\partial \boldsymbol{x}$ has a term involving the variance and the third derivative of the function. Let us call that term as *drift term*. In an ideal case scenario, the derivative of the objective function with respect to the parameter $\partial f/\partial \boldsymbol{x}$ should be equal to the derivative of the convoluted objective function with respect to the mean of the distribution $\partial U/\partial \boldsymbol{\mu}$. Let the optimum parameter obtained using the gradient descent algorithm performed on U be $\boldsymbol{\theta}^* = \{\boldsymbol{\mu}^*, \boldsymbol{\sigma}^*\}$ and on f be $\boldsymbol{x}^*$. If we ensure that $\partial f/\partial \boldsymbol{x} = \partial U/\partial \boldsymbol{\mu}$, the $\boldsymbol{\mu}^*$ and $\boldsymbol{x}^*$ should converge to the same value. This is achievable when either $\|\boldsymbol{\sigma}\| \to 0$ or the third derivative term tends to zero.

Now, taking the derivative of Eq. (17) with respect to $\boldsymbol{\sigma}$ and ignoring the higher order terms of $\boldsymbol{\sigma}$, we get

$$\frac{\partial U}{\partial \sigma_i} = \sigma_i \frac{\partial f^2}{\partial^2 x_i}. \tag{19}$$

The double derivative is positive at the local minimum. If the function does not have a very high rate of change of curvature, $\partial U/\partial \sigma_i > 0$ in the proximity of the local minimum. This means that the gradient descent method decreases the value of $\|\boldsymbol{\Sigma}\|$ as one comes close to the local minimum. A smaller value of the $\|\boldsymbol{\Sigma}\|$ leads to a smaller drift term. This means that if the optimizer reaches the proximity of a local minimum, the derivative of the convoluted objective function(U) with respect to the mean of the distribution($\boldsymbol{\mu}$) comes closer to the derivative of the actual objective function(f) with respect to the parameter space($\boldsymbol{x}$). So, the optimizer converges to the local minimum of the objective function.

2.6.2 Gradient Correction for constraints

Let us consider an optimization problem:

$$\min_x x^2, \quad \text{s.t} \quad 1 - x \leq 0. \tag{20}$$

The optimum for the optimization problem in Eq. (20) lies at 1.0, which coincides with the boundary of the feasible domain. The Fig. 3 shows the plot corresponding to the constraint element of the augmented function $\mathcal{L}(x)$, as well as the constraint segment within the upper bound $U(\mu, \sigma)$ over a range of variance values for the distribution. Ideally, the constraint term should not exert influence within the confines of the feasible region. The Fig. 3 reveals that the constraint term's influence on $U(\mu, \sigma)$ remains nonzero within the feasible zone. This influence progressively diminishes with the reduction in the σ value. Notably, this influence tends to deter the optimizer from maintaining proximity to the constraint boundary, even when the optimum resides along this boundary. When we run our suggested algorithm to solve the problem in Eq. (20), the algorithm will reach the area inside the feasible boundary and get pushed away from the boundary because of the constraints. Eventually, the value of σ decreases, and the effect of the constraint around the boundary diminishes,

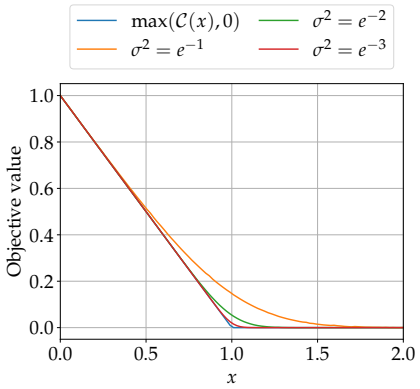
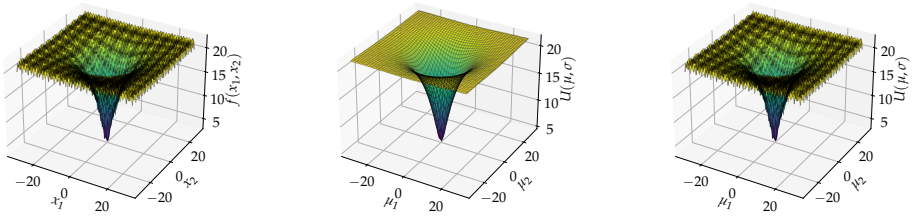


Figure 3: Contribution of the constraint term for the problem given in Eq. (20). The blue curve shows the constraint term in the augmented function $\mathcal{L}(x)$. The orange, green, and red curves show the constraint term in the upper bound $U(\mu, \sigma)$ for decreasing values of the σ. We observe that the tendency of the constraint to move the optimum away from the boundary is higher for a bigger value of σ. This misguides the optimizer, especially when the optimum lies on the boundary of the feasible region.

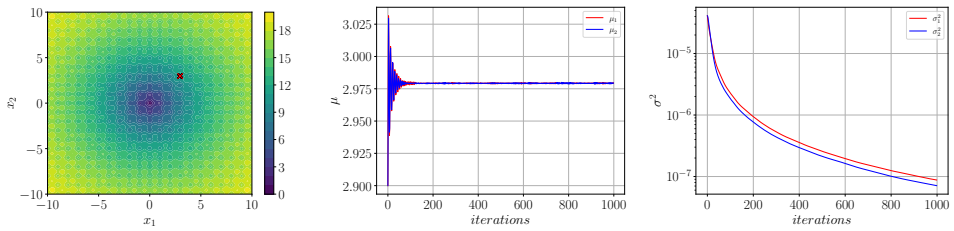
moving towards convergence at the boundary. This approach is suboptimal, mainly when dealing with computationally demanding objective functions, as it decelerates the convergence rate. We propose a strategy aimed at mitigating this effect by enforcing the value of the gradient of the constraint's contribution to the upper bound (U) with respect to the position parameter ($\boldsymbol{\mu}$) to assume a zero value if that constraint is satisfied $\Big($i.e., $\frac{\partial U_{\mathcal{C}_j}}{\partial \boldsymbol{\mu}} = 0, \quad \text{if} \quad \mathcal{C}_j(\boldsymbol{\mu}) \leq 0, \text{ where } U_{\mathcal{C}_j} = \mathbb{E}_{\boldsymbol{x},\boldsymbol{b}}[\mathcal{C}_j(\boldsymbol{x}, \boldsymbol{b})]\Big)$.

2.6.3 Overcoming multiple local optima



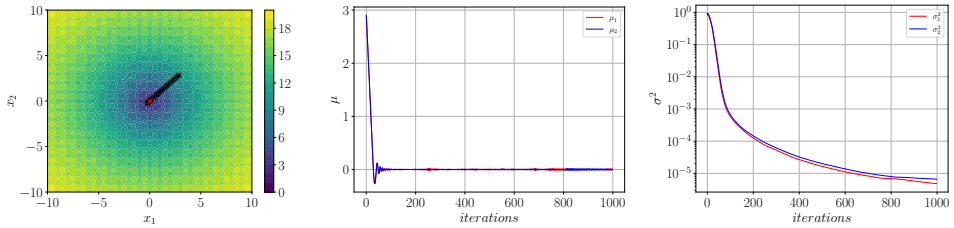
(a) 2D Ackley function as $f(\boldsymbol{x})$ (b) $U(\boldsymbol{\theta})$ with $\boldsymbol{\Sigma} = e^{0}\boldsymbol{I}$ (c) $U(\boldsymbol{\theta})$ with $\boldsymbol{\Sigma} = e^{-10}\boldsymbol{I}$

Figure 4: Comparing the effect of Gaussian convolution on the 2D Ackley function.



(a) Mean evolution plotted over $f(\boldsymbol{x})$ value. (b) Mean evolution (c) Variance evolution

Figure 5: Effect of starting the optimization with a very small variance. The starting value of variance is given as $\boldsymbol{\Sigma} = \text{diag}(\sigma_1^2, \sigma_2^2)$ with $\sigma_1^1 = \sigma_2^2 = e^{-10}$.



(a) Mean evolution plotted over $f(\boldsymbol{x})$ value.

(b) Mean evolution

(c) Variance evolution

Figure 6: Effect of starting the optimization with a high variance. The starting value of variance is given as $\boldsymbol{\Sigma} = \text{diag}(\sigma_1^2, \sigma_2^2)$ with $\sigma_1^1 = \sigma_2^2 = e^0$.

Casting the objective $f(\boldsymbol{x})$ as $U(\boldsymbol{\theta})$ has the additional advantage of escaping local optima by smoothing the objective function. It can be viewed as a Gaussian-blurred version of the original objective $f(\cdot)$, free of non-smoothness. The degree of this smoothing is contingent upon the choice of $\boldsymbol{\Sigma}$. Examplalarily, we demonstrate this smoothening effect using the Ackley function, which has a highly non-convex surface, as illustrated in Fig. 4. The figure hints towards the significant effect the initial value of the design variable variance can have on the quality of the optimum. We optimized the 2D Ackley function using our proposed algorithm under two distinct initial variance scenarios to study the effect of initial design variable variance. This is illustrated in Fig. 5 and Fig. 6 for small and high variance, respectively. The optimizer is trapped in a local minimum when the initial variance is small (ref. Fig. 5) and dodges several local optima to converge at the global optima (ref. Fig. 6) when the initial variance is high. This property is also confirmed upon testing using the Rastringen function (ref. Fig. 21). These tests underscore the importance of starting with considerably significant variance.

2.6.4 Sample Size

Let us assume that after k^{th} step of optimization, we are in a state $\boldsymbol{\theta}_k := (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, where $\boldsymbol{\mu}_k := \{\mu_{1,k}, \ldots, \mu_{d,k}\}$ and $\boldsymbol{\Sigma}_k := \text{diag}(\sigma_{1,k}^2, \ldots, \sigma_{d,k}^2)$. Let us represent the exact and the Monte Carlo approximation of the gradient in Eq. (7) by $\nabla_{\boldsymbol{\theta}_k} U$ and $\widehat{\nabla_{\boldsymbol{\theta}_k} U}$ respectively. Let us represent the variance of the samples used to approximate the gradient in Eq. (7) at the k^{th} optimization step by $\boldsymbol{V}_k := \{V_{1,k}, \ldots, V_{d,k}, \ldots, V_{2d,k}\}$ with

$$V_{i,k} = \mathbb{V}\left[\mathcal{L}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda}) \frac{\partial}{\partial \theta_i} \log q\left(\boldsymbol{x} \mid \boldsymbol{\theta}_k\right)\right]. \tag{21}$$

One can write the MSE of the gradient approximation as

$$\mathbb{E}\left[\left(\widehat{\nabla_{\boldsymbol{\theta}_k} U} - \nabla_{\boldsymbol{\theta}_k} U\right)^2\right] = \frac{\boldsymbol{V}_k}{S}. \tag{22}$$

Upon using the vanilla stochastic gradient descent method with learning rate α, the next exact state ($\boldsymbol{\theta}_{k+1}$), and the approximate state ($\widehat{\boldsymbol{\theta}}_{k+1}$) can be obtained as

$$\begin{aligned} \boldsymbol{\theta}_{k+1} &= \boldsymbol{\theta}_k + \alpha \nabla_{\boldsymbol{\theta}_k} U, \\ \widehat{\boldsymbol{\theta}}_{k+1} &= \boldsymbol{\theta}_k + \alpha \widehat{\nabla_{\boldsymbol{\theta}_k} U}. \end{aligned} \tag{23}$$

Note that this analysis can be extended to other variations of the gradient descent methods by changing the formula in Eq. (23). We can write the MSE of the state as

$$\mathbb{E}\left[\left(\widehat{\boldsymbol{\theta}}_{k+1} - \boldsymbol{\theta}_{k+1}\right)^2\right] = \alpha^2 \frac{\boldsymbol{V}_k}{S}. \tag{24}$$

The Monte Carlo estimator in Eq. (7) is unbiased i.e., $\mathbb{E}[\widehat{\nabla_{\boldsymbol{\theta}_k} U}] = \nabla_{\boldsymbol{\theta}_k} U$. This transfers the unbiasedness to the states, which leads to

$$\mathbb{E}\left[\widehat{\boldsymbol{\theta}}_{k+1}\right] = \boldsymbol{\theta}_{k+1}. \tag{25}$$

Now, we want to choose the number of samples such that distance between the Gaussian distribution obtained by exact gradient value ($q(\boldsymbol{\theta}_{k+1})$) and the Gaussian distribution obtained by the approximate gradient ($q(\widehat{\boldsymbol{\theta}}_{k+1})$) in probability space is less than a desired value given by ε_{KL}. This distance measure is given by the Kullback-Leibler divergence [72], written as

$$
\begin{aligned}
D_{\mathrm{KL}}(q(\widehat{\boldsymbol{\theta}}_{k+1}) \mid\mid q(\boldsymbol{\theta}_{k+1})) &= \frac{1}{2}\left(\log\frac{\mid \boldsymbol{\Sigma} \mid}{\mid \widehat{\boldsymbol{\Sigma}} \mid} + tr\left(\boldsymbol{\Sigma}^{-1}\widehat{\boldsymbol{\Sigma}}\right) + (\widehat{\boldsymbol{\mu}} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\widehat{\boldsymbol{\mu}} - \boldsymbol{\mu}) - d\right), \\
&= \frac{1}{2}\sum_{i=1}^{d}\left(\log\sigma_{i,k+1}^2 - \log\widehat{\sigma}_{i,k+1}^2 + \frac{\widehat{\sigma}_{i,k+1}^2}{\sigma_{i,k+1}^2} + \frac{(\widehat{\mu}_{i,k+1} - \mu_{i,k+1})^2}{\sigma_{i,k+1}^2}\right) - \frac{d}{2},
\end{aligned}
\tag{26}
$$

where $\widehat{(\cdot)}$ represent the states obtained using approximate gradients. Taking expectation on both sides of Eq. (26), then simplifying the expression using Eq. (24), and Eq. (25), we obtain

$$
\begin{aligned}
&\mathbb{E}\left[D_{\mathrm{KL}}(q(\widehat{\boldsymbol{\theta}}_{k+1}) \mid\mid q(\boldsymbol{\theta}_{k+1}))\right] \\
&= \frac{1}{2}\sum_{i=1}^{d}\left(\log\sigma_{i,k+1}^2 - \mathbb{E}\left[\log\widehat{\sigma}_{i,k+1}^2\right] + \frac{\mathbb{E}\left[\widehat{\sigma}_{i,k+1}^2\right]}{\sigma_{i,k+1}^2} + \frac{\mathbb{E}\left[(\widehat{\mu}_{i,k+1} - \mu_{i,k+1})^2\right]}{\sigma_{i,k+1}^2}\right) - \frac{d}{2}, \\
&= \frac{1}{2}\sum_{i=1}^{d}\left(\log\sigma_{i,k+1}^2 - \mathbb{E}\left[\log\widehat{\sigma}_{i,k+1}^2\right] + \frac{\alpha^2 V_{i,k}}{S\sigma_{i,k+1}^2}\right), \\
&< \frac{1}{2}\sum_{i=1}^{d}\left(\log\sigma_{i,k+1}^2 - \underbrace{\log\mathbb{E}\left[\widehat{\sigma}_{i,k+1}^2\right]}_{\text{Jensen's ineqality}} + \frac{\alpha^2 V_{i,k}}{S\sigma_{i,k+1}^2}\right) = \sum_{i=1}^{d}\frac{\alpha^2 V_{i,k}}{2S\sigma_{i,k+1}^2} \quad \text{(using Eq. (25))}, \\
\implies &\mathbb{E}\left[D_{\mathrm{KL}}(q(\widehat{\boldsymbol{\theta}}_{k+1}) \mid\mid q(\boldsymbol{\theta}_{k+1}))\right] < \sum_{i=1}^{d}\frac{\alpha^2 V_{i,k}}{2S\sigma_{i,k+1}^2}
\end{aligned}
\tag{27}
$$

We obtained the upper bound of the KL divergence between the approximate and exact distribution. We enforce a bound on the KL divergence by bounding the upper bound by a parameter $\varepsilon_{\mathrm{KL}}$, providing an expression for the sample size

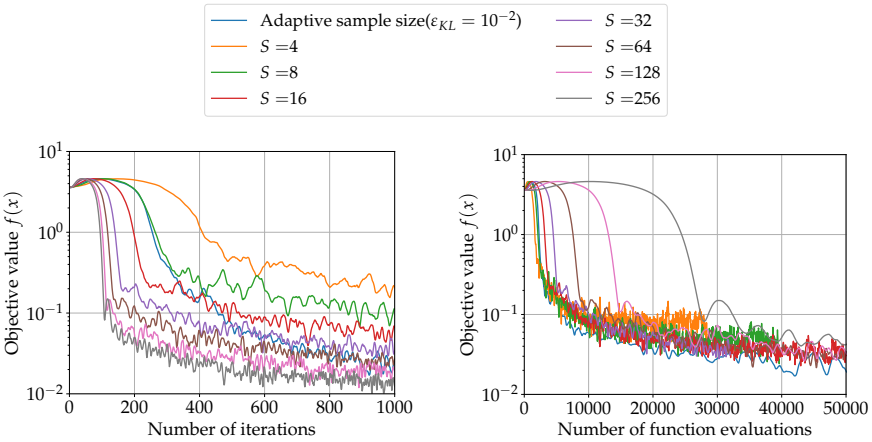
$$
\begin{aligned}
&\mathbb{E}\left[D_{KL}(q(\widehat{\boldsymbol{\theta}}_{k+1}) \mid\mid q(\boldsymbol{\theta}_{k+1}))\right] < \varepsilon_{\mathrm{KL}}, \\
\implies &\sum_{i=1}^{d}\frac{\alpha^2 V_{i,k}}{2S\sigma_{i,k+1}^2} < \varepsilon_{\mathrm{KL}}, \\
\implies &S > \sum_{i=1}^{d}\frac{\alpha^2 V_{i,k}}{2\varepsilon_{\mathrm{KL}}\sigma_{i,k+1}^2}.
\end{aligned}
\tag{28}
$$

When one uses natural gradients then the expression becomes

$$
S > \sum_{i=1}^{d}\frac{\alpha^2 V_{i,k}\sigma_{i,k}^4}{2\varepsilon_{\mathrm{KL}}\sigma_{i,k+1}^2}. \tag{29}
$$

From the above, we observe:

- For higher dimensional problems, more samples are required.
- Variance reduction techniques help in decreasing sample size requirement by reducing the variance of the gradient $\boldsymbol{V}_k$.
- The number of samples is proportional to the step size. Smaller step sizes require fewer samples because of small changes in the parameter values in the gradient descent step.
- When employing natural gradients, the number of samples required is relatively smaller when closer to the optimum (i.e., $\sigma_{i,k}^4 \to 0$).



(a) Evolution of the objective function value against the number of optimization steps

(b) Evolution of the objective function value against the number of function evaluations

Figure 7: Numerical study to compare the performance SCOUT-Nd with adaptive sample size and fixed sample sizes on 32-dimensional Ackley function. We switch off the natural gradients and QMC for all the cases to ensure a similar condition for all the test cases.

By using Eq. (28) or Eq. (29), we can dynamically determine the number of samples required at each optimization step. We initially sample a predefined number of points and then verify if the conditions are met. If they are unsatisfied, we sample additional points to fulfill the requirements. We iteratively add the newly sampled points, continuing this process until the conditions in Eq. (28) and Eq. (29) are satisfied. To conserve computational resources, we utilize both new and existing points instead of resampling all points. However, this approach precludes using QMC due to the non-nested behavior of Sobol points.

The value of $\mathbf{\Sigma}_{k+1}$ is not known at the beginning of the k^{th} optimization step. To address this, we make the assumption that the change from $\mathbf{\Sigma}_k$ to $\mathbf{\Sigma}_{k+1}$ is relatively small but has a significant impact on Eq. (28). First, we compute the number of samples by substituting $\sigma^2_{i,k+1}$ with $\sigma^2_{i,k}$ in Eq. (28). Then, we calculate $\mathbf{\Sigma}_{k+1}$ through a gradient descent step and verify if the conditions for Eq. (28) are met. If they are not, we adjust the number of samples and recalibrate the value of $\mathbf{\Sigma}_{k+1}$ by repeating the gradient descent step. This process is iterated until Eq. (28) is satisfied.

We numerically evaluate the performance of adaptive sample size against various fixed sample size cases in optimizing the 32-dimensional Ackley function. Figure 7 illustrates the evolution of the objective function against the number of optimization steps and function evaluations. As discussed in Algorithms 2 and 1, natural gradients are employed when the variance falls below a specified threshold ($\sigma_{\text{nat-grad}}$). Since natural gradients start at different steps for different test cases, they are excluded from this numerical test to ensure fair comparison. Additionally, we do not use Quasi-Monte Carlo (QMC) methods for fixed sample size cases, as QMC is incompatible with the adaptive sample size method.

From Fig. 7(a), we observe that the convergence rate improves as the number of samples increases. Additionally, the optimum quality is better with a larger sample size. This is attributed to the smaller mean square error(MSE) in gradient estimation associated with larger sample sizes, leading to faster convergence and improved optimum results. Notably, optimization with adaptive sample sizes falls in between, achieving an optimum quality comparable to that of optimization with 128 samples.

We observe from Fig. 7(b) that as the number of samples increases, the convergence slows down. While maintaining a low mean squared error (MSE) in each iteration brings advantages, it also leads to more function evaluations. Optimization using a small sample size converges faster in terms of function evaluations. However, the optimum quality is compromised due to the high noise in gradient estimation, resulting in oscillations around the optimum. In contrast, optimization with adaptive sample size selects an appropriate number of samples based on the estimator's variance.

Based on these observations, we can conclude that a higher sample size is suitable when function evaluation is inexpensive or we have sufficient computational resources to parallelize the gradient estimation step. However, in cases where function evaluation is expensive, using an adaptive sample size is more practical.

2.7 Multi-fidelity

The main computational bottleneck of the gradient estimation using Eq. (7) is the multiple evaluation of the objective function. This becomes a significant concern for computationally expensive simulators. We propose to solve this problem using the multi-fidelity (MF) method [73] in the Scout algorithm, termed as MF-SCOUT-Nd. Suppose we are given a set of L functions modeling the same quantity and arranged in ascending order of accuracy and computational cost $\{f^{(1)}, f^{(2)}, \ldots, f^{(L)}\}$ and the corresponding augmented functions as $\{\mathcal{L}^{(1)}, \mathcal{L}^{(2)}, \ldots, \mathcal{L}^{(L)}\}$. To evaluate the gradients, we are calculating the expectation. Following the methods mentioned in [74], we can write the highest fidelity augmented function as a telescopic sum of the other fidelities:

$$\begin{aligned}\mathcal{L}^{(L)}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda}) &= \mathcal{L}^{(1)}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda}) + (\mathcal{L}^{(2)}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda}) - \mathcal{L}^{(1)}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda})) + \\ &\quad \ldots + (\mathcal{L}^{(L)}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda}) - \mathcal{L}^{(L-1)}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda})) \\ &= \mathcal{L}^{(1)}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda}) + \sum_{\ell=2}^{L} \left(\mathcal{L}^{(\ell)}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda}) - \mathcal{L}^{(\ell-1)}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda}) \right). \end{aligned} \tag{30}$$

Multiplying with $\nabla_{\boldsymbol{\theta}} \log q(x|\boldsymbol{\theta})$ and taking expection on both the sides:

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{b}} \left[\mathcal{L}^{(L)} \nabla_{\boldsymbol{\theta}} \log q(x|\boldsymbol{\theta}) \right] = \mathbb{E}_{\boldsymbol{x},\boldsymbol{b}} \left[\mathcal{L}^{(1)} \nabla_{\boldsymbol{\theta}} \log q(x|\boldsymbol{\theta}) \right] + \sum_{\ell=2}^{L} \mathbb{E}_{\boldsymbol{x},\boldsymbol{b}} \left[(\mathcal{L}^{(\ell)} - \mathcal{L}^{(\ell-1)}) \nabla_{\boldsymbol{\theta}} \log q(x|\boldsymbol{\theta}) \right]. \tag{31}$$

As per the Eq. (6), the left-hand side of the equation is the gradient of the highest fidelity function with respect to the distribution parameter $\left(\nabla_{\boldsymbol{\theta}} U^{(L)}(\boldsymbol{\theta})\right)$. All the expectations on the right-hand side of the equation can be estimated using an unbiased Monte Carlo estimator. Now, the equation above is simplified as

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} U^{(L)}(\boldsymbol{\theta}) \approx &\frac{1}{S_1} \sum_{i=1}^{S_1} \mathcal{L}^{(1)}(\boldsymbol{x}_i, \boldsymbol{b}_i, \boldsymbol{\lambda}) \frac{\partial}{\partial \boldsymbol{\theta}} \log q\left(\boldsymbol{x}_i \mid \theta\right) \\ &+ \sum_{\ell=2}^{L} \frac{1}{S_\ell} \sum_{i=1}^{S_\ell} \left(\mathcal{L}^{(\ell)}(\boldsymbol{x}_i, \boldsymbol{b}_i, \boldsymbol{\lambda}) - \mathcal{L}^{(\ell-1)}(\boldsymbol{x}_i, \boldsymbol{b}_i, \boldsymbol{\lambda}) \right) \frac{\partial}{\partial \boldsymbol{\theta}} \log q\left(\boldsymbol{x}_i \mid \theta\right), \end{aligned} \tag{32}$$

where S_ℓ is the number of samples used in the estimator at the fidelity level ℓ. We assume that the approximation quality between each fidelity improves as we increase the fidelity. So, the number of samples required to approximate the expectation decreases as the fidelity increases (i.e., $S_1 > S_2 > \ldots > S_L$). We replace Eq. (3) and Eq. (7) with Eq. (30) and Eq. (32) respectively in the Algorithm.1 and 2, with the rest of the steps remaining the same to handle multi-fidelity.

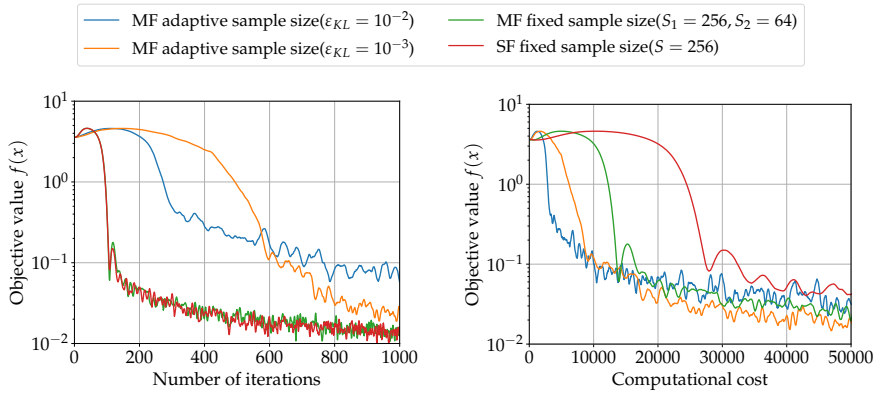
2.7.1 Sample size for multi-fidelity derivative estimator

In this section, we will discuss the number of samples allocated to each fidelity level for the estimation of the derivative. We follow the approach and notations used in Sec. 2.6.4. Let us define the variable $V_k^{(\ell)}$ as

$$V_k^{(\ell)} = \begin{cases} \sum_{i=1}^{d} \frac{\mathbb{V}\left[\mathcal{L}^{(1)}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda}) \nabla_{\theta_i} \log q(\boldsymbol{x}|\boldsymbol{\theta}_k)\right]}{2\sigma_{i,k+1}^2}, & \text{if } \ell = 1, \\ \sum_{i=1}^{d} \frac{\mathbb{V}\left[\left(\mathcal{L}^{(\ell)}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda}) - \mathcal{L}^{(\ell-1)}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda})\right) \nabla_{\theta_i} \log q(\boldsymbol{x}|\boldsymbol{\theta}_k)\right]}{2\sigma_{i,k+1}^2}, & \text{otherwise.} \end{cases} \tag{33}$$

Following the steps in Sec. 2.6.4, we can derive the following for the multi-fidelity case

$$\mathbb{E}\left[D_{\mathrm{KL}}(q(\widehat{\boldsymbol{\theta}}_{k+1}) \mid\mid q(\boldsymbol{\theta}_{k+1})) \right] < \sum_{\ell=1}^{L} \frac{\alpha^2 V_k^{(\ell)}}{S_\ell}. \tag{34}$$



(a) Evolution of the objective function value against the number of optimization steps.

(b) Evolution of the objective function value against the computational cost.

Figure 8: Numerical study to compare the performance MF-SCOUT-Nd with adaptive sample size and fixed sample sizes on 32-dimensional Ackley function (the function defined in Appendix. C). We switch off the natural gradients for all the cases to ensure a similar condition for all the test cases. We assume the high-fidelity model is four times more expensive than the low-fidelity model.

Let C_ℓ be the cost of evaluating $f^{(\ell)}$. We obtain the number of samples for each level by minimizing the total cost such that the upper bound of the KL divergence in Eq. (34) is equal to ε_{KL}. This approach is extensively used in multilevel Monte Carlo literature [74, 75, 76]. The optimization problem is stated as

$$\begin{aligned} S^*_{\ell,k} &= \arg\min_{S_\ell} \sum_{\ell=1}^{L} C_\ell S_\ell, \\ \text{s.t.} \quad & \sum_{\ell=1}^{L} \frac{\alpha^2 V_k^{(\ell)}}{S_\ell} = \varepsilon_{\text{KL}}, \end{aligned} \tag{35}$$

where $S^*_{\ell,k}$ represents the optimum number of samples for the ℓ^{th} fidelity and k^{th} step. One can analytically solve this problem using a Lagrange multiplier. The optimum number of samples is

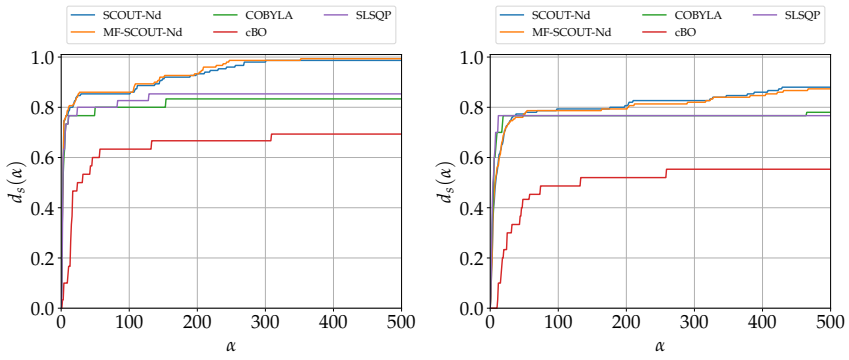
$$S^*_{\ell,k} = \left(\frac{\sum_{\ell=1}^{L} \sqrt{V_k^{(\ell)} C_\ell}}{\varepsilon_{\text{KL}}} \right) \alpha^2 \sqrt{\frac{V_k^{(\ell)}}{C_\ell}}. \tag{36}$$

The value of $\boldsymbol{\Sigma}_{k+1}$ is not known when we want to calculate the number of samples. To overcome this problem, we use the same procedure as described in Sec. 2.6.4.

We conducted a numerical evaluation to compare the performance of multi-fidelity gradient estimation using fixed and adaptive sample sizes against single-fidelity fixed sample size cases with 256 samples to optimize the 32-dimensional Ackley function. Figure 8 illustrates the evolution of the objective function against the number of optimization steps and function evaluations. We did not employ natural gradient and QMC methods for the same reasons discussed in Sec. 2.6.4. We assume the high-fidelity model is four times more expensive than the low-fidelity model.

From Fig. 8(a), we observe that the convergence rate with respect to the optimization step for the multi-fidelity method with fixed sample size is almost the same as the single-fidelity methods with 256 samples even when multi-fidelity method uses less computational resources demonstrating the advantage of using the multi-fidelity method. Additionally, the multi-fidelity adaptive sample size approach with a smaller upper bound ($\varepsilon_{\text{KL}} = 10^{-3}$) reaches a better the optimum as compared to the bigger upper bound ($\varepsilon_{\text{KL}} = 10^{-2}$) due to a more accurate approximation of the gradients.

We observe from Fig. 8(b) that multi-fidelity methods converge faster with respect to the computational cost than the single-fidelity method. The multi-fidelity adaptive sample method also outperforms the multi-fidelity fixed sample size method by selecting the optimal number of points based on the estimator's variance while minimizing the computational cost. The case with a larger



(a) $\epsilon_f = 0.1$ (b) $\epsilon_f = 0.01$

Figure 9: Data Profile plots to show the aggregate performance of the different optimization problems on the given set of benchmark problems. We observe that SCOUT-Nd and MF-SCOUT-Nd were able to solve most of the benchmark problems with both accuracy levels $\epsilon_f = 0.1$, and $\epsilon_f = 0.01$.

ε_{KL} exhibits a faster initial convergence rate but results in a lower quality optimum. Therefore, it is advisable to begin with a larger value of ε_{KL} for faster initial convergence and decrease it as you approach the optimum.

Based on these observations, we conclude that the multi-fidelity methods help reduce computational resources compared to the single-fidelity method, and the adaptive formulation further decreases the computational load. Additionally, one can enhance the optimum quality by adjusting the value of ε_{KL}.

3 Numerical Illustrations

This section compares the proposed algorithm with the state-of-the-art algorithms using standard benchmark analytical problems. Additionally, we highlight the algorithm's efficacy in a complex real-world application: optimizing wind farm layout. In the following, we use `PyTorch` [77] to efficiently compute the gradient of the densities. After the gradient estimation, we use the ADAM optimizer [78] as the stochastic gradient descent method. In this work, $q(\boldsymbol{x} \mid \boldsymbol{\theta})$ takes the form of a Gaussian distribution unless otherwise stated with parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ representing mean and diagonal covariance[2], respectively. The code for the algorithm and the numerical experiments can be found in https://github.com/KislayaRavi/scout-Nd.

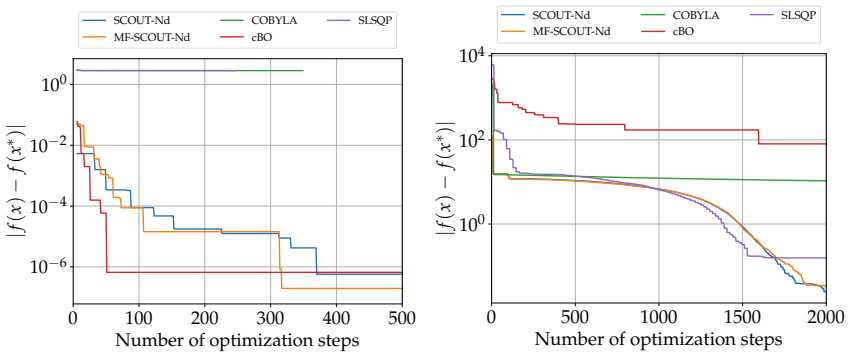
3.1 Benchmark Studies

Before applying our algorithm to a real-world problem, we test it on standard optimization benchmarks. We select thirty benchmarks to test the algorithm's ability to handle different types of optimization challenges like multi-modality, constraints, behavior in valleys, dimension scalability, etc. The list of benchmarks is given in the Appendix C. We slightly modify the function to make the corresponding low-fidelity function for MF-SCOUT-Nd. For each problem, Gaussian noise with a standard deviation of 0.01 was added to turn the problem stochastic.

We compare our proposed algorithm with state-of-the-art derivative-free optimization algorithms like Constrained Optimization by Linear Approximation (COBYLA) [46], Sequential Least SQuares Programming (SLSQP) [47], and Constrained Bayesian Optimization (cBO) [45]. We use the implementation of COBYLA and SLSQP from the Scipy library [79] and the implementation of cBO from the BayesianOptimization library [80]. We run 1000 optimization steps for each optimizer.

We use the data profile curve [29] to compare the overall performance of the algorithms for the whole set of benchmarks. Let $\mathcal{S}$ represent the set of optimizers and $\mathcal{P}$ be a set of benchmark problems. The

[2]In the subsequent investigation $\boldsymbol{\Sigma} = \text{diag}(\sigma_1^2, \ldots, \sigma_d^2)$ with $\sigma_i^2 = e^{2\beta_i}$ unless otherwise stated.



(a) 3-dimensional Hartmann function (b) 16-dimensional Rosenbrock function

Figure 10: Evolution of optimum for 3-dimensional Hartmann function and 16-dimensional Rosenbrock function. 3-dimensional Hartmann function has 4 local minima. We observe from Figure 10(a) that COBYLA and SLSQP got stuck in a local minimum, whereas other methods that can handle multi-modal problems reached a global minimum. cBO struggles in high-dimensional problems such as 16-dimensional Rosenbrock functions as depicted in Figure 10(b)

.

Table 1: The table showing the number of low and high-fidelity function evaluations (S_{LF} and S_{HF}) for different benchmark problem (p) as a function of the dimension (d_p)

	$d_p \leq 2$		$2 < d_p \leq 4$		$4 < d_p \leq 8$		$d_p > 8$	
	S_{LF}	S_{HF}	S_{LF}	S_{HF}	S_{LF}	S_{HF}	S_{LF}	S_{HF}
Single-fidelity case	-	32	-	64	-	128	-	256
Multi-fidelity case	32	8	64	16	128	32	256	64

data profile $d_s(\alpha)$ [29] of an optimizer $s \in \mathcal{S}$ is given by

$$d_s(\alpha) = \frac{1}{|\mathcal{P}|} \left| \left\{ p \in \mathcal{P} : \frac{t_{p,s}}{d_p + 1} \leq \alpha \right\} \right|, \tag{37}$$

where $p \in \mathcal{P}$ represents a benchmark problem, d_p is the dimension of benchmark and $t_{p,s}$ is the minimum number of optimization steps a solver s requires to reach the optimum of a problem p within accuracy level ϵ_f. We ran each benchmark five times, each with different seeds. So, the total number of benchmarks is $|\mathcal{P}|= 30$(number of unique benchmarks) $\times$ 5(number of random seeds) $= 150$. In the benchmark studies, we use a fixed sample size for expectation evaluation for all the optimization methods to ensure an unbiased comparison. The number of low-fidelity samples (S_{LF}) and high-fidelity samples (S_{HF}) to calculate the expectation at each step of optimization depends upon the dimension of the benchmark problem as shown in Table 1.

We can observe from Fig. 9 that SCOUT-Nd and MF-SCOUT-Nd solved more benchmark problems with a given level of accuracy(ϵ_f). Our proposed algorithms outperform other derivative-free methods because we use gradient approximation to move toward the optimum. This not only helped us to converge faster but also tackled high-dimensionality. MF-SCOUT-Nd had a similar performance level as SCOUT-Nd, but it uses less high-fidelity function evaluations to reach the same level of accuracy, thereby saving computational resources.

Our experiments show that COBYLA and SLSQP struggle to handle multimodal models. In some benchmark cases, these two methods converge early to local optima. For instance, we observe this behavior for the 3-dimensional Hartmann function, which has 4 local minima. COBYLA and SLSQP get stuck in one of the local minima as observed in Fig. 10(a).

Bayesian optimization methods are known to struggle in high-dimensional space [81]. Moreover, they also struggle in the Rosenbrock function, where it finds the valley but takes a long time to converge to the minimum [81]. We plot the evolution of the optimum for the 16-dimensional Rosenbrock function in Fig. 10(b) and observe that cBO struggles to handle high-dimensionality and valleys.

3.2 Windfarm Layout Optimization

We apply SCOUT-Nd and MF-SCOUT-Nd to windfarm layout optimization problem [15]. The primary goal of wind farm layout optimization is to position the wind turbines to reduce interference and thus maximize power production. This is a challenging and complex problem due to the variable nature of the wind and the complex interactions between turbines through their wakes. The challenges are listed as follows:

- Uncertainty due to the wind. Generally, the wind speed and wind direction are treated as random variables.
- Availability of only black box evaluations of the numerical simulators [48, 9]. This would make gradient-based methods rely on finite differences, which would significantly increase derivative computational cost and derivatives may also have significant errors [82]. If gradient-free methods are employed, they may struggle when the problem is high dimensional as they do not usually scale well to problems with more than 30 design variables [83].
- A highly multimodal design space, making the problem highly non-convex [84]. This multimodality makes it difficult for classical gradient-based methods, as they require the objective function to be smooth enough [15], or else they might get stuck in local optimum. Gradient-free methods can handle multimodality better, however, they suffer from the curse of dimensionality as discussed above. Thus, there exists no best algorithm, the choice of which is situation and problem-dependent [9, 48].

Owing to the challenges discussed, and their significant overlap with the strengths of the proposed algorithm, the problem serves as an ideal testing ground for SCOUT-Nd/MF-SCOUT-Nd.

Problem Definition: The objective of the wind farm layout optimization is to maximize the AEP [3] by changing the position of the wind turbines. The turbines are constrained to stay within a given area and with a minimum separation between them. This objective and the constraints result in a problem of nonlinear optimization under uncertainty with deterministic constraints:

$$\arg\min_{\boldsymbol{x}} \mathbb{E}_{\boldsymbol{b}}[-AEP(\boldsymbol{x}, \boldsymbol{b})] \quad \text{s.t} \quad \boldsymbol{\mathcal{C}}(\boldsymbol{x}) \leq 0, \tag{38}$$

where $\boldsymbol{b} = \{b_1, b_2, \ldots, b_n\}$ represents a vector of the random variable for the wind data with $p(\boldsymbol{b})$ being the joint probability density function of the uncertain variables. Common uncertain variables are the wind direction and the freestream wind speed [15]. $\boldsymbol{x}$ denotes a sequence of pairs of coordinates for each wind turbine, which can take on continuous values in a bounded domain. The expected AEP is computed by marginalizing the uncertain parameters. To marginalize, some commonly used methods are Monte Carlo, Polynomial Chaos, rectangular quadrature, etc to name a few [85]. In this work, we used a weighted average, which amounts to the rectangular integration rule. $\boldsymbol{\mathcal{C}}(\boldsymbol{x}) = \{\mathcal{C}_1(\boldsymbol{x}), \mathcal{C}_2(\boldsymbol{x})\}$ denotes the constraints this problem has with $\mathcal{C}_1(\boldsymbol{x})$ denoting the separation between the two turbines constraint and $\mathcal{C}_2(\boldsymbol{x})$ denoting the area constraint. In particular, they are defined as

$$\mathcal{C}_1(\boldsymbol{x}) = \mathcal{K}(2D - Q_{i,j}) \quad i, j = 1 \ldots n_{turbines}; \;\; i \neq j \tag{39}$$

$$\mathcal{C}_2(\boldsymbol{x}) = \mathcal{M}(N_{i,m}) \quad i = 1 \ldots n_{turbines}; \;\; m = 1 \ldots n_{boundaries}, \tag{40}$$

where $Q_{i,j}$ is the distance between each pair of turbines i and j , and D is the turbine diameter. The normal distance, $N_{i,m}$, from each turbine i to each boundary m is defined as negative when a turbine is inside the boundary and positive when it is outside the boundary. We aggregate the distance between two turbine constraints using the Kreisselmeier–Steinhauser functional $\mathcal{K}(\cdot)$[86], which reduces the number of constraints to 1. Also, we define a function $\mathcal{M}(\cdot)$, which aggregates the normal distance constraints by taking the mean of the positive values of $N_{i,m}$. $\mathcal{M}(\cdot)$ is calculated as

$$\mathcal{M}(\cdot) = \frac{1}{|N^+|} \sum_{\eta_i \in N^+} \eta_i; \qquad \text{with, } \; N^+ = \{\eta_i \in N | \eta_i > 0\} \tag{41}$$

[3] The Annual energy production (AEP) is given by expected power multiplied by the number of hours in a year.

where $|N^+|$ is the cardinality of the set N^+. These aggregations produce a less complex optimization problem.

Implementation details/test cases In the experiments performed, we used the NREL 5MW reference turbine [87]. For wake models, we have used their implementation in FLOw Redirection and Induction in Steady State (FLORIS) [88]. We use Jensen [89][4] as the low-fidelity (LF) wake model and Gauss-Curl Hybrid (GCH) [90][5] as the high-fidelity (HF) model as suggested in [91]. The low- and high-fidelity models for this problem use different *wind roses*[6] bin resolutions, leading to accuracy and computational differences caused by both fidelity and resolution. Each bin adds an identical set of function calls and operations to the summing process, so the cost scales linearly. In the subsequent experiments, the wind speed is constant at $8m/s$ for simplicity.

We investigate two cases with different numbers of wind turbines, as shown in Table 2. The particular choice is made to study the influence of dimensions of the design variable. We use six wind direction bins for the LF and 18 wind direction bins for the HF (ref. Fig. 11). In terms of computational costs, a single LF model evaluation, denoted as c_{LF}, incurs approximately 33% (or one-third) of the cost associated with a single evaluation of the HF model, represented as c_{HF}. For example, in the scenario involving 24 turbines, the Jensen model (the LF model) requires approximately 0.015 seconds per execution. In contrast, the GCH model(HF model) demands about 0.044 seconds per execution when run on a single processor core. Consequently, the overall computational expense of determining the Annual Energy Production (AEP) using the HF model is about nine times higher than that required for the LF model.

For both cases, we perform a comparative study between SCOUT-Nd with the HF model, MF-SCOUT-Nd employing both the LF and the HF model, and the Sequential Least SQuares Programming (SLSQP) [47] with the HF model. Within the domain of wind farm layout optimization, variants of sequential quadratic programming (SQP)-based algorithms are predominantly employed [85, 9, 91] due to their gradient-based nature (gradients computed via finite differences) and their capability to manage constraints effectively. An extensive comparison with different optimization methods is beyond the scope of the present investigation. Interested readers can find details in [9].

Table 2: Optimization problem cases: design variables, objective, and constraints

Cases	objective	design variables	dimension	Constraints
8 turbine farm	AEP	x, y locations	16	Turbine spacing $[252\text{ m}, \infty)$, bound constraint ($x \in [0, 2666\text{ m}], y \in [0, 2666\text{ m}]$)
24 turbine farm	AEP	x, y locations	48	Turbine spacing $[252\text{ m}, \infty)$, bound constraint ($x \in [0, 8000\text{ m}], y \in [0, 8000\text{ m}]$)

Results This section presents the results obtained upon running the optimization methods for the two cases presented in Table. 2. The hyper-parameter setting for the optimization methods is given in Appendix D, particularly in Table. 5 and Table. 6 for the 8 turbines and 24 turbine cases, respectively. Here, we use a fixed sample size for expectation evaluation for all the optimization methods to ensure a fair comparison. For both cases, a grid layout is chosen as the initial layout for all three optimization methods. Naturally, the optimizer is expected to push the turbines towards the boundaries to minimize wake interactions, leading to increased AEP. The results for the two cases with the optimization methods are quantitatively compared in Table. 3 and Table. 4, and qualitatively compared in Fig. 12, Fig. 16, Fig. 13 and Fig. 18. The evolution of the augmented objective, objective and constraints for the optimization with SCOUT-Nd and MF-SCOUT-Nd for 8 turbine case are illustrated in Fig. 14

[4]The Jensen wake model uses a simplistic velocity deficit to represent the wake, and this deficit is summed when wakes interact using the sum-of-squares method.

[5]The solver simplifies the Reynolds-averaged Navier-Stokes (RANS) equations to obtain a parabolic equation for the wake deficit. The equation is solved in a three-dimensional domain to obtain the wake velocity in a wind plant.

[6]Wind rose is a graphical tool used in the context of wind farms to represent the distribution of wind speed and direction at a particular location

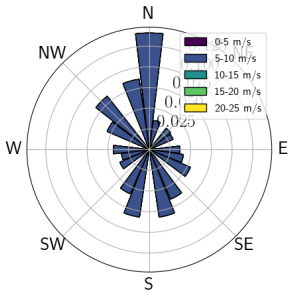


Figure 11: Wind Rose for 18 wind direction bins and a constant wind speed of 8 m/s

and Fig. 15 respectively. Similarly, Fig. 19 and Fig. 20 illustrate the evolutions for the 24 turbine case.

We draw the following conclusions from the tables and figures:

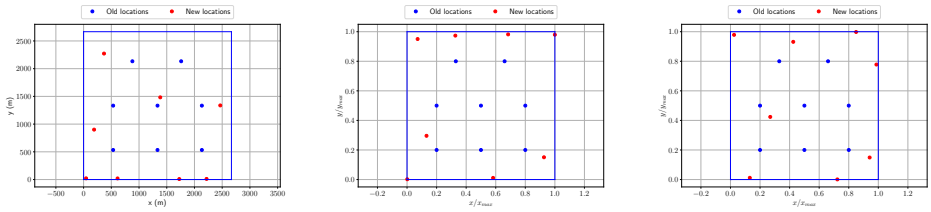
- For the 8 turbine case, from Fig. 12 we observe that all three optimization methods push the turbine towards the boundaries while maintaining the distance between them as per the constraints. This significantly reduces the wake interactions as observed in Fig. 13. This translates to an increase in AEP as reported in Table. 3. In this case, SLSQP reports the highest AEP percentage gain (8.73%) from the initial grid layout, followed by SCOUT-Nd (8.26%) and MF-SCOUT-Nd (8.02%). However, the MF-SCOUT-Nd is the cheapest with around 67% computational cost of the SLSQP. This was expected as the SLSQP is a gradient-based method, and dim=16 poses a fairly smooth objective surface, outperforming method like ours. With an optimized value of hyperparameters, SCOUT-Nd/MF-SCOUT-Nd might come closer (or even cross) to the SLSQP in terms of AEP improvement, but this is not the goal of the present investigation.
- For the 24 turbine case, from Fig. 16 we observe that both SCOUT-Nd and MF-SCOUT-Nd push the turbine towards the boundaries from the initial grid layout, which is expected from global optima. This can also be understood from the power vs. wind direction plot in Fig. 17. Additionally, Fig. 18 illustrates the wake interactions along the most prominent wind direction, i.e., from the north side. However, as seen in the Fig. 16, the SLSQP failed to push the turbine towards the boundaries as it got trapped in local optima. We also confirm this behavior by running the SLSQP optimizer using different initial layouts drawn from the Latin hypercube as presented in Fig. 22. With the increase in dimensions, owing to the multimodel nature of the problem, the gradient-based optimizers can face this issue, also reported in [9, 84, 15]. This is also reflected in the AEP values reported in the Table. 4. The SCOUT-Nd reports the highest AEP percentage gain from the initial grid layout, followed by MF-SCOUT-Nd and then the SLSQP. Although the MF-SCOUT-Nd resulted in slightly lower AEP gain as compared to the SCOUT-Nd, interestingly, it took around 1/3 of the cost of SCOUT-Nd optimization with HF. This underscores the capability of our method to be robust and to handle non-convexity in high dimensions (as asserted in Sec. 2.6) in real-world physical applications. By introducing Multi-fidelity, our method allows the analyst to balance the trade-off between the computational cost and the optimum quality.
- The Figures.14 15, 19 and 20 point towards the convergence of SCOUT-Nd and MF-SCOUT-Nd for both the cases by illustrating the evolution of the augmented objective, objective and constraints. As can be seen from the figures, initially the constraints on average are violated. As the optimization progresses, the penalty is also more strongly enforced, thus satisfying the constraints eventually. Further diagnostics of the optimization, like the evolution of the design variables and penalty parameters, are given in Appendix D. In all the cases presented, the optimization is terminated by the ϵ_σ convergence criterion for the outer loop. The optimum quality can be further increased by reducing its value, although with an increased computational cost. The convergence criterion, number of samples, and step size are entwined together in a complicated fashion, influencing the convergence rate and the optimum quality as discussed in Sec. 2.6.4 and Sec. 2.7.1. As with any optimization

algorithm [34], we rely on carefully selected heuristics to balance the tradeoff between computational cost and solution accuracy.

Table 3: Optimization results comparison for 8 turbine case.

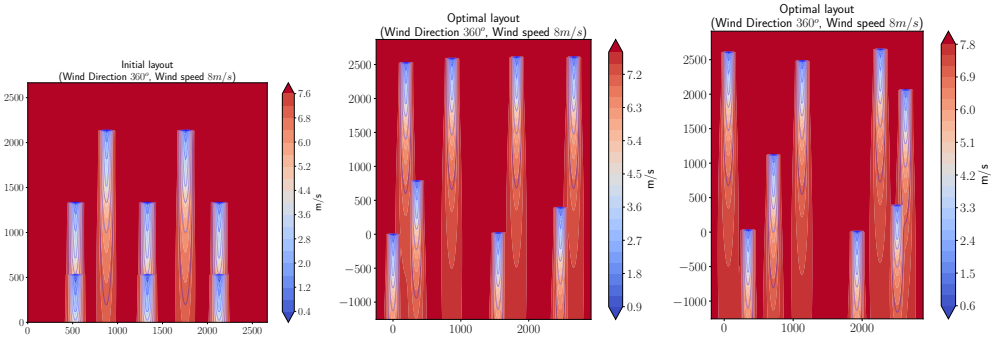
Algo.	LF calls	HF calls	Computational cost (in c_{HF}[#])	AEP (in MWh)[†]	% AEP gain[†]
SCOUT-Nd	-	$600 \cdot 32 \cdot 18$[*]	$345,600 c_{HF}$ (1.83x)	$116,472.50$	8.26%
MF-SCOUT-Nd	$615 \cdot 32 \cdot 6$[*]	$615 \cdot 8 \cdot 18$[*]	$\mathbf{127,526.4 c_{HF}}$ (0.67x)	$116,214.37$	8.02%
SLSQP	-	$10,484 \cdot 18$	$188,712 c_{HF}$ (1x)	$116,978.31$	8.73%

[†] The initial gird layout AEP = $107,581.42$ MWh.
[#] c_{HF} is the cost of a high-fidelity run for a single pair of wind direction and wind speed. $c_{LF} \approx 0.33 c_{HF}$.
[*] total no. of steps $\times$ no. of samples/step $\times$ no. of solver call/AEP evaluation.



(a) SLSQP (b) SCOUT-Nd (c) MF-SCOUT-Nd

Figure 12: Windfarm layout with 8 turbines comparing the layout used to initialize (in blue) the optimization routine and the optimized layout (in red) found using SLSQP (a), SCOUT-Nd (b) and MF-SCOUT-Nd (c). The SLSQP and SCOUT-Nd use the GCH model [90], while the MF-SCOUT-Nd uses a combination of the less accurately resolved Jensen wake model [89] and the GCH model. Each dot represents a wind turbine. The blue line represents the boundary of the wind farm. Quantitative Results are given in Table. 3 and the hyperparameters used to perform the optimization are detailed in Table. 5.

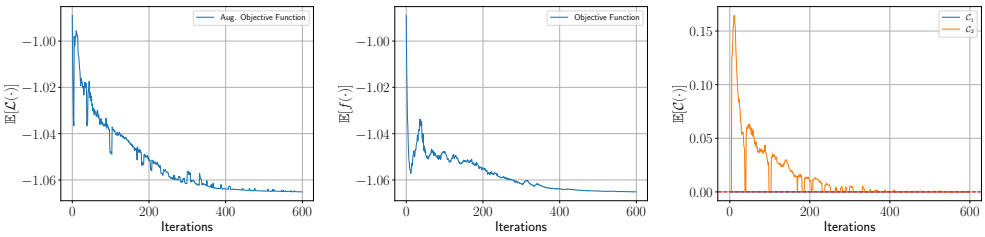


(a) Initial Layout (b) SCOUT-Nd (c) MF-SCOUT-Nd

Figure 13: Velocity deficit contour plots along the primary wind direction for the initial layout (a), optimized layout with SCOUT-Nd (b), and optimized layout with MF-SCOUT-Nd (c). The corresponding AEPs are given in Table. 3.

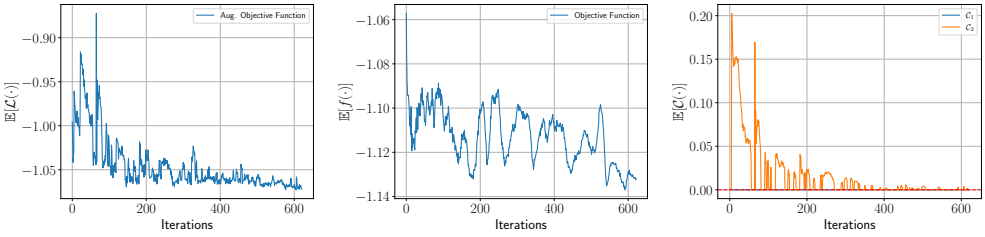
4 Conclusions

We presented SCOUT-Nd, a novel approach for (constrained) optimization problems involving expensive, stochastic black-box simulators with high-dimensional parametric dependency. We



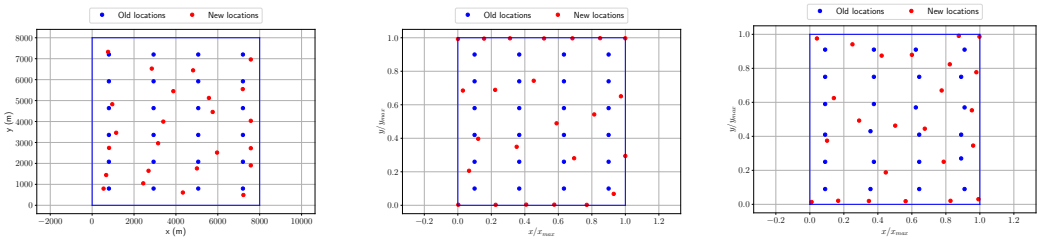
(a) (b) (c)

Figure 14: Diagnostics of the optimization run with SCOUT-Nd involving the high-fidelity GCH model [90], for the 8 turbine wind farm case. From left to right, the expected value of augmented objective (a), objective (b), and constraints (c). The objective, constraints, and design variables are normalized. The physical meanings are given in Table.2.



(a) (b) (c)

Figure 15: Diagnostics of the optimization run with MF-SCOUT-Nd involving the low-fidelity Jensen wake model [89] and the high-fidelity GCH model [90], for the 8 turbine wind farm case. From left to right, the expected value of augmented objective (a), objective (b), and constraints (c). The objective, constraints, and design variables are normalized. The physical meanings are given in Table.2.



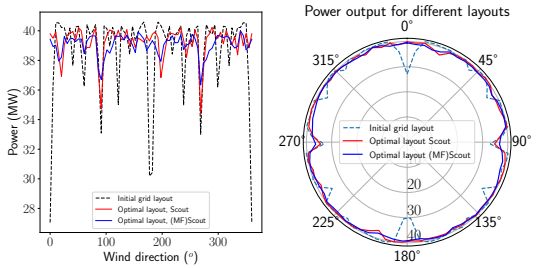
(a) SLSQP (b) SCOUT-Nd (c) MF-SCOUT-Nd

Figure 16: Windfarm layout with 24 turbines comparing the layout used to initialize (in blue) the optimization routine and the optimized layout (in red) found using SLSQP (a), SCOUT-Nd (b) and MF-SCOUT-Nd (c). The SLSQP and SCOUT-Nd use the GCH model [90], while the MF-SCOUT-Nd uses a combination of the less accurately resolved Jensen wake model [89] and the GCH model. Each dot represents a wind turbine. The blue line represents the boundary of the wind farm. Quantitative Results are given in Table. 4 and the hyperparameters used to perform the optimization are detailed in Table. 6.

Table 4: Optimization results comparison for 24 turbine case.

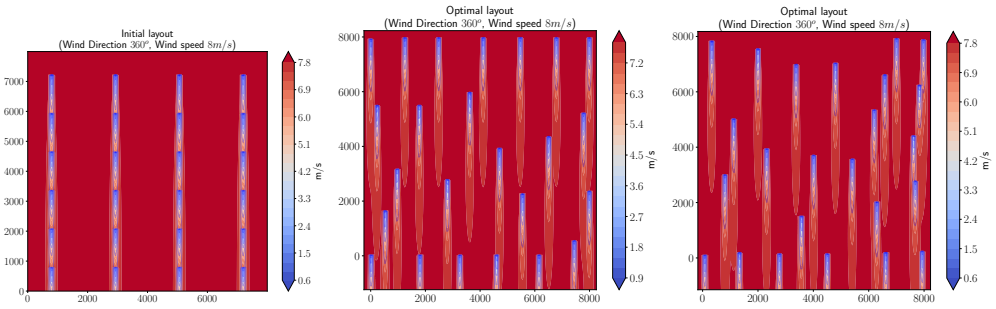
Algo.	LF calls	HF calls	Computational cost (in c_{HF}[#])	AEP (in MWh)[†]	% AEP gain[†]
SCOUT-Nd	-	$2,648 \cdot 128 \cdot 18^{*}$	$6,100,992 c_{HF}$ (3.13x)	$\mathbf{347,112.88}$	3.98%
MF-SCOUT-Nd	$2,350 \cdot 128 \cdot 6^{*}$	$2,350 \cdot 32 \cdot 18^{*}$	$\mathbf{1,949,184 c_{HF}}$ (1x)	$343,662.69$	2.98%
SLSQP	-	$7,200 \cdot 18$	$129,600 c_{HF}$[††]	$336,221.53$	0.71%

[††] The optimization run gets trapped in local minima around iteration 120, hence the low AEP and computational cost..
[†] The initial gird layout AEP = $333,834.84$ MWh.
[#] c_{HF} is the cost of a high-fidelity run for a single pair of wind direction and wind speed. $c_{LF} \approx 0.33 c_{HF}$.
[*] total no. of steps $\times$ no. of samples/step $\times$ no. of solver call/AEP evaluation.



(a) (b)

Figure 17: Wind farm power as a function of wind direction for the initial grid layout, SCOUT-Nd optimized layout, and the MF-SCOUT-Nd optimized layout for the 24 turbine case.

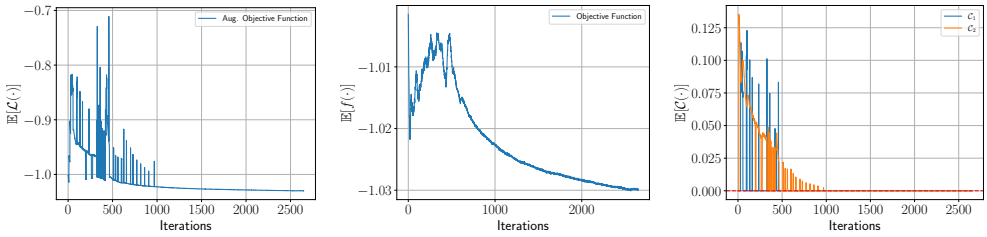


(a) Initial layout (b) SCOUT-Nd (c) MF-SCOUT-Nd

Figure 18: Velocity deficit contour plots along the primary wind direction (from the North side) for the initial layout (a), optimized layout with SCOUT-Nd (b), and optimized layout with MF-SCOUT-Nd (c). The corresponding AEPs are given in Table. 4.

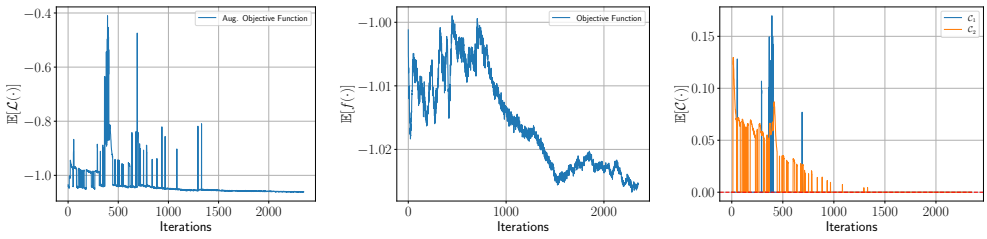
included strategies and carefully chosen heuristics to handle non-convexity, reduce gradient estimator variance, and improve convergence properties and robustness. We also extended the proposed approach to include multi-fidelity strategies and adaptive sample size for gradient estimation to limit the number of expensive to evaluate simulator calls. Our method is embarrassingly parallelizable and non-intrusive, thus very attractive to various engineering and physics optimization problems.

We performed experiments on various academic toy problems with common optimization challenges like multi-modality, constraints, behavior in valleys, dimension scalability, etc., and compared our algorithm against several baselines. For a given level of accuracy and computational budget, our method solves most of the problems from the toy problem pool. We also demonstrated that it could handle multi-modalities and high-dimensionality better than the baselines. Additionally, we demonstrated the improved quality of the optimum and better convergence rate when including



(a) (b) (c)

Figure 19: Diagnostics of the optimization run with SCOUT-Nd involving the high-fidelity GCH model [90], for the 24 turbine wind farm case. From left to right, the expected value of augmented objective (a), objective (b), and constraints (c). The objective, constraints, and design variables are normalized. The physical meanings are given in Table.2.



(a) (b) (c)

Figure 20: Diagnostics of the optimization run with MF-SCOUT-Nd involving the low-fidelity Jensen wake model [89] and the high-fidelity GCH model [90], for the 24 turbine wind farm case. From left to right, the expected value of augmented objective (a), objective (b), and constraints (c). The objective, constraints, and design variables are normalized. The physical meanings are given in Table.2.

natural gradients. We also tested our algorithm on a complex real-world case of optimizing wind farm layout and compared it against a baseline. We observed that our method improved upon the baseline optimization results, thus showing the successful optimization of a complex stochastic system with a user-defined objective function. Also, the MF-SCOUT-Nd produced comparable accuracy at a one-third cost.

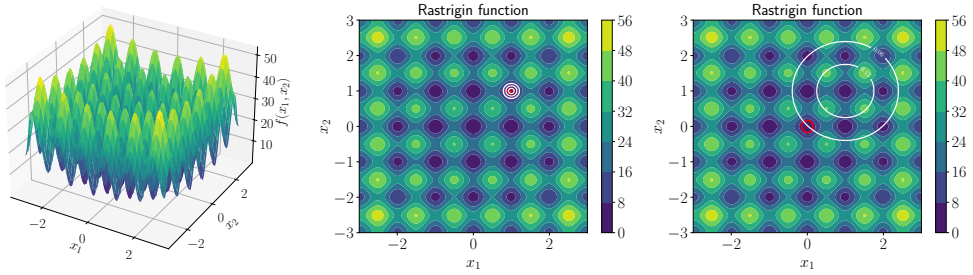
The work presented serves as a fertile ground for several extensions and applications. In future research, we intend to adapt and apply the proposed method to optimize hyperparameters in neural networks. Since the algorithm addresses the differentiability issue in problems involving physics-based simulators, integrating into several Scientific Machine Learning paradigms is also possible [92, 25]. This includes hybrid approaches that combine "known physics" (physics-based simulators) and "learned physics" (neural network, for example), thus relying on efficient gradient flow. As suggested in [34], we plan to incorporate a full covariance matrix into the design variable density to enhance performance. Furthermore, integrating Importance sampling step into gradient estimation could yield improvements, particularly in scenarios where the objective function landscape contains valleys.

Acknowledgements

A.A. and P.S.K acknowledge the support of VDI Technologiezentrum Gmbh and the Federal Ministry for Education and Research (BMBF) in the context of the project "Probabilistic machine learning for the calibration and validation of physics-based models of concrete" (FKZ-13XP5125B). K.R and H.J.B acknowledge the support of Helmholtz Association under the research school Munich School for Data Science (MUDS). The authors declare that there is no conflict of interest regarding the publication of this paper.

Appendix

A Influence of initial value of design variable variance



(a) Rastringin function surface (b) Initial value of $\boldsymbol{\Sigma} = e^{-2}\boldsymbol{I}$ (c) Initial value of $\boldsymbol{\Sigma} = e^{0}\boldsymbol{I}$

Figure 21: Using 2d Rastringin function to highlight the influence of the initial value of design variance on escaping local optima. White contour lines represent the initial $q(\boldsymbol{\theta})$ and red contour lines represent the final design variable distribution $q(\boldsymbol{\theta}^*)$

B Taylor Expansion simplification

Our proposed algorithm relies on optimizing $U(\boldsymbol{\theta})$ instead of $f(\boldsymbol{x})$. Let us convert the distribution to standard normal distribution as follows:

$$U(\boldsymbol{\theta}) = \mathbb{E}[f(\boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{z})]_{\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})}. \tag{42}$$

Writing down the Taylor expansion of $f(\boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{z})$ till the third derivative:

$$f(\boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{z}) = f(\boldsymbol{\mu}) + \sum_{i=1}^{d} \sigma_i z_i \frac{\partial f}{\partial x_i}(\boldsymbol{\mu}) + \frac{1}{2!} \sum_{i,j=1}^{d} \sigma_i \sigma_j z_i z_j \frac{\partial^2 f}{\partial x_i \partial x_j}(\boldsymbol{\mu}) \dots$$
$$+ \frac{1}{3!} \sum_{i,j,k=1}^{d} \sigma_i \sigma_j \sigma_k z_i z_j z_k \frac{\partial^3 f}{\partial x_i \partial x_j \partial x_k}(\boldsymbol{\mu}) + O(\|\boldsymbol{\sigma}\|^4),$$

where z_i, σ_i and x_i are the i^{th} component of $\boldsymbol{z}$, $\boldsymbol{\sigma}$ and $\boldsymbol{x}$ respectively. We take expectation on both sides to get $U(\boldsymbol{\theta})$.

$$\begin{aligned} U(\boldsymbol{\theta}) &= \mathbb{E}[f(\boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{z})]_{\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \\ &= f(\boldsymbol{\mu}) + \sum_{i=1}^{d} \sigma_i \mathbb{E}[z_i] \frac{\partial f}{\partial x_i}(\boldsymbol{\mu}) + \frac{1}{2!} \sum_{i,j=1}^{d} \sigma_i \sigma_j \mathbb{E}[z_i z_j] \frac{\partial^2 f}{\partial x_i \partial x_j}(\boldsymbol{\mu}) \dots \\ &+ \frac{1}{3!} \sum_{i,j,k=1}^{d} \sigma_i \sigma_j \sigma_k \mathbb{E}[z_i z_j z_k] \frac{\partial^3 f}{\partial x_i \partial x_j \partial x_k}(\boldsymbol{\mu}) + O(\|\boldsymbol{\sigma}\|^4). \end{aligned} \tag{43}$$

We simplify the above expression using the following observations:

$$\mathbb{E}[z_i] = 0 \quad \text{because of 0 mean}$$
$$\mathbb{E}[z_i z_j] = \begin{cases} 1 & \text{if } i = j \text{ because standard deviation of the distribution is 1} \\ 0 & \text{otherwise because the variables are independent and have zero mean} \end{cases}$$
$$\mathbb{E}[z_i z_j z_k] = 0 \quad \text{because skewness of normal distribution is zero}$$

The simplified expression is:

$$U(\boldsymbol{\theta}) = f(\boldsymbol{\mu}) + \frac{1}{2} \sum_{i=1}^{d} {\sigma_i}^2 \frac{\partial^2 f}{\partial {x_i}^2}(\boldsymbol{\mu}) + O(\|\boldsymbol{\sigma}\|^4) \tag{44}$$

C List of benchmarks

We represent the dimension of the search space by d_p. The list of benchmarks is as follows:

C.1 Sphere problem

We run this problem for dimensions $d_p = \{2, 4, 8, 16, 32\}$. The high-fidelity and the low-fidelity function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= \sum_{i=1}^{d} x_i^2, \\ f_{\text{low}}(x) &= \sum_{i=1}^{d} 1.1 x_i^2. \end{aligned} \tag{45}$$

It is a simple problem with a single global minimum at the $(0, \ldots d_p \text{ times})$.

C.2 Constrained Sphere problem

We run this problem for dimensions $d_p = \{2, 4, 8, 16, 32\}$. The objection function is same as Equation 45 but with a constraint $1 - x_1 - x_2 \leq 0$. The optimum lies on the constraint boundary at $(0.5, 0.5, 0, \ldots d_p - 2 \text{ times})$. The problem checks the ability of the algorithm to handle constraints.

C.3 Ackley function

We run this problem for dimensions $d_p = \{2, 4, 8, 16, 32\}$. The high-fidelity and the low-fidelity function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= -20 \exp\left(-0.2\sqrt{\frac{1}{d_p}\sum_{i=1}^{d_p} x_i^2}\right) - \exp\left(\frac{1}{d_p}\sum_{i=1}^{d_p}\cos(2\pi x_i)\right) + 20 + \exp(1), \\ f_{\text{low}}(x) &= -22 \exp\left(-0.2\sqrt{\frac{1}{d_p}\sum_{i=1}^{d_p} 1.1 x_i^2}\right) - 0.9 \exp\left(\frac{1}{d_p}\sum_{i=1}^{d_p}\cos(2\pi x_i)\right) + 20 + \exp(1). \end{aligned} \tag{46}$$

It is a multimodal problem with a global minimum at $(0, \ldots d \text{ times})$.

C.4 Rosenbrock function

We run this problem for dimensions $d_p = \{2, 4, 8, 16\}$. The high-fidelity and the low-fidelity function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= \sum_{i=1}^{d_p - 1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2, \\ f_{\text{low}}(x) &= \sum_{i=1}^{d_p - 1} 101(x_{i+1} - x_i^2)^2 + 1.01(1 - x_i)^2 + 0.2. \end{aligned} \tag{47}$$

The function is unimodal, but the minimum lies in a narrow valley is at $(1, 1, \ldots d_p \text{ times})$. The optimizers find the valley of the problem but take a long time to converge to the minimum [81].

C.5 Zakharov function

We run this problem for dimensions $d_p = \{2, 4, 8, 16\}$. The high-fidelity and the low-fidelity function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= \sum_{i=1}^{d_p} x_i^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{d_p} 0.5ix_i\right)^4, \\ f_{\text{low}}(x) &= \sum_{i=1}^{d_p} x_i^2 + \left(\sum_{i=1}^{d} 0.55ix_i\right)^2 + \left(\sum_{i=1}^{d_p} 0.5ix_i\right)^4. \end{aligned} \tag{48}$$

It has one minimum at $(0, \ldots d_p \text{ times})$. It is a plate-shaped function. Optimizers tend to get stuck in the flat regions of the function.

C.6 Bohachevsky function: 1

This is a 2-dimensional bowl-shaped problem. The high-fidelity and the low-fidelity function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7, \\ f_{\text{low}}(x) &= x_1^2 + 2.1x_2^2 - 0.32\cos(3\pi x_1) - 0.41\cos(4\pi x_2) + 0.7. \end{aligned} \tag{49}$$

The minimum lies at $(0, 0)$.

C.7 Bohachevsky function: 2

This is a 2-dimensional bowl-shaped problem. The high-fidelity and the low-fidelity function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3, \\ f_{\text{low}}(x) &= x_1^2 + 2.1x_2^2 - 0.31\cos(3\pi x_1)\cos(4\pi x_2) + 0.3. \end{aligned} \tag{50}$$

The minima lies at $(0, 0)$.

C.8 Six-hump camel function

This is a 2-dimensional function with six local minima. The high-fidelity and the low-fidelity function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= 4x_1^2 - 2.1x_1^4 + \frac{x_1^6}{3} + x_1x_2 - 4x_2^2 + 4x_2^4, \\ f_{\text{low}}(x) &= 4x_1^2 - 2.2x_1^4 + \frac{x_1^6}{3.2} + 1.1x_1x_2 - 4.1x_2^2 + 4x_2^4. \end{aligned} \tag{51}$$

There are two global minima at $(0.0898, -0.7126)$ and $(0.0898, 0.7126)$.

C.9 Three-hump camel function

This is a 2-dimensional function with three local minima. The high-fidelity and the low-fidelity function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2, \\ f_{\text{low}}(x) &= 2.1x_1^2 - 1.06x_1^4 + \frac{x_1^6}{6} + 1.1x_1x_2 + x_2^2. \end{aligned} \tag{52}$$

C.10 Beale function

This is a 2-dimensional multimodal problem. The high-fidelity and the low-fidelity function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2, \\ f_{\text{low}}(x) &= (1.54 - x_1 + x_1x_2)^2 + (2.29 - x_1 + x_1x_2^2)^2 + (2.675 - x_1 + x_1x_2^3)^2. \end{aligned} \tag{53}$$

The minima lies at $(3, 0.5)$.

C.11 Hartmann 3d function

This is a 3-dimensional problem with four local minima. The high-fidelity and the low-fidelity function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2\right), \\ f_{\text{low}}(x) &= -1.1\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2\right) + 0.1, \end{aligned} \tag{54}$$

where α, A and P are defined as:

$$\begin{aligned} \alpha &= (1, 1.2, 3, 3.2)^T \\ A &= \begin{pmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix} \\ P &= \begin{pmatrix} 0.3689 & 0.1170 & 0.2673 \\ 0.4699 & 0.4387 & 0.7470 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.0381 & 0.5743 & 0.8828 \end{pmatrix} \end{aligned}$$

The global minima lies at $(0.114614, 0.555649, 0.852547)$.

C.12 Hartmann 4d function

This is a multimodal 4-dimensional problem. The high-fidelity and the low-fidelity function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{4} A_{ij}(x_j - P_{ij})^2\right), \\ f_{\text{low}}(x) &= -1.1\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{4} A_{ij}(x_j - P_{ij})^2\right) + 0.1, \end{aligned} \tag{55}$$

where α, A and P are defined as:

$$\begin{aligned} \alpha &= (1, 1.2, 3, 3.2)^T \\ A &= \begin{pmatrix} 10 & 3 & 17 & 3.5 \\ 0.05 & 10 & 17 & 0.1 \\ 3 & 3.5 & 1.7 & 10 \\ 17 & 8 & 0.05 & 10 \end{pmatrix} \\ P &= \begin{pmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 \end{pmatrix} \end{aligned}$$

The global minima lies at $(0.1873, 0.1906, 0.5566, 0.2647)$.

D Further details of the windfarm layout optimization

In the windfarm layout optimization case, we observed that for high dimensional optimization, the SLSQP was trapped in local optima when optimization was started from a grid layout. To confirm this

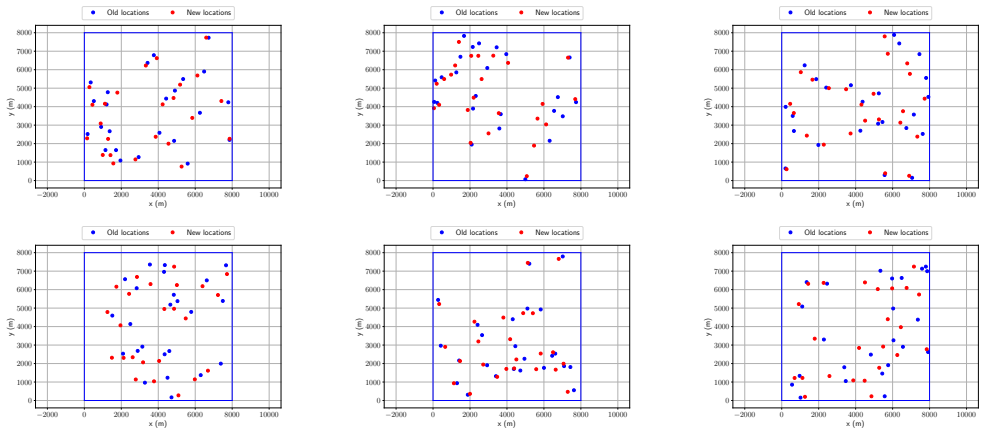


Figure 22: Windfarm layout optimization with SLSQP for 24 turbine case with 6 different initial layouts sampled with latin hypercube. In the six runs, the final layout never reaches the boundaries (as expected for an optimal layout) and the optimizer gets trapped in the local optima

behavior, we ran the SLSQP optimizer using different initial layouts drawn from the latin hypercube as presented in Fig. 22.

Table 5 and Table 6 show the hyperparameters of the algorithms used for the windfarm layout optimization case for 8 turbine and 24 turbine cases, respectively.

The Figures.23 24, 25 and 26 provides optimization diagnostics of SCOUT-Nd and MF-SCOUT-Nd for both the cases. In all four figures, it is interesting to observe the normalized design variable mean fluctuation near the wind farm boundary around the initial iterations and the corresponding constraint violation. Gradually as the penalty is more strongly enforced (by increasing the penalty parameter λ) the constraints are satisfied on average, accompanied by stabilization of the design variable mean.

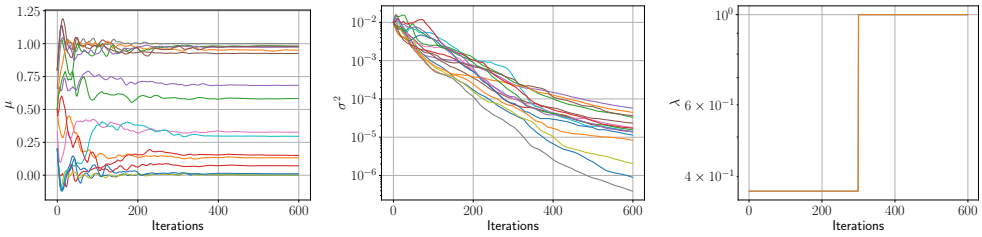
Table 5: Hyper-parameters of algorithms used to carry-out experiments for 8 turbine windfarm case.

SCOUT-Nd		MF-SCOUT-Nd		SLSQP†	
parameter	value	parameter	value	parameter	value
η	$5e-02$	η	$5e-02$	`ftol`	$1e-10$
S_{HF}	32	(S_{LF}, S_{HF})	(32,8)	`maxiter`	400
ε_μ	$1e-05$	ε_μ	$1e-05$	`eps`	0.01
ε_σ	$1e-04$	ε_σ	$1e-04$	-	-

† The parameters of SLSQP correspond to the parameters in its Scipy implementation. More details can be found here and [47].

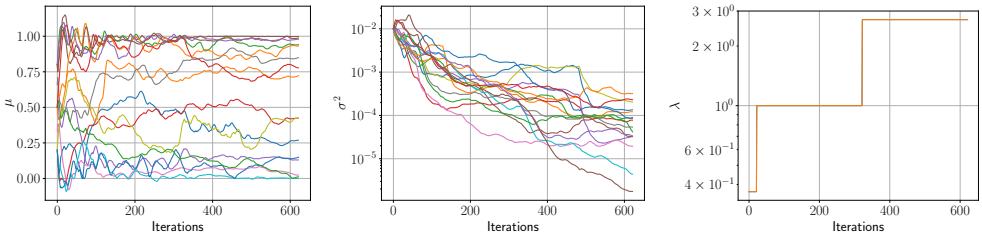
Table 6: Hyper-parameters of algorithms used to carry-out experiments for 24 turbine windfarm case

SCOUT-Nd		MF-SCOUT-Nd		SLSQP	
parameter	value	parameter	value	parameter	value
η	$1e-02$	η	$1e-02$	`ftol`	$1e-12$
S	128	(S_{LF}, S_{HF})	(128,32)	`maxiter`	400
ε_μ	$1e-05$	ε_μ	$1e-05$	`eps`	0.01
ε_σ	$5e-04$	ε_σ	$5e-04$	-	-



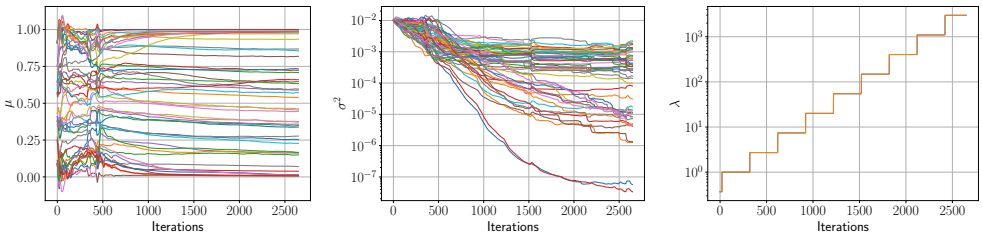
(a) (b) (c)

Figure 23: Diagnostics of the optimization run with SCOUT-Nd involving the high-fidelity GCH model [90], for the 8 turbine wind farm case. From left to right, the mean of the design variables (a), the variance of the design variables (b), and the penalty term λ (c).



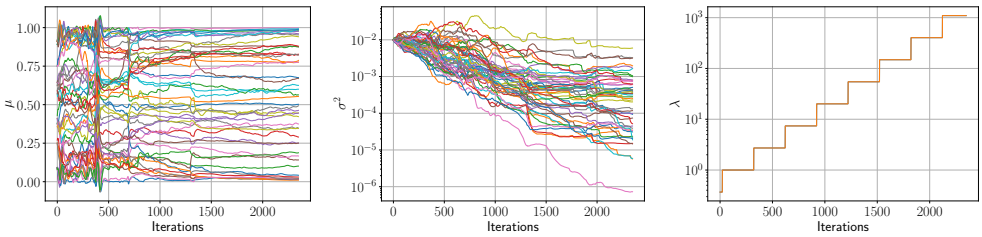
(a) (b) (c)

Figure 24: Diagnostics of the optimization run with MF-SCOUT-Nd involving the high-fidelity GCH model [90], for the 8 turbine wind farm case. From left to right, the mean of the design variables (a), the variance of the design variables (b), and the penalty term λ (c).



(a) (b) (c)

Figure 25: Diagnostics of the optimization run with SCOUT-Nd involving the high-fidelity GCH model [90], for the 24 turbine wind farm case. From left to right, the mean of the design variables (a), the variance of the design variables (b), and the penalty term λ (c).



(a) (b) (c)

Figure 26: Diagnostics of the optimization run with MF-SCOUT-Nd involving the high-fidelity GCH model [90], for the 24 turbine wind farm case. From left to right, the mean of the design variables (a), the variance of the design variables (b), and the penalty term λ (c).

References

[1] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. "The frontier of simulation-based inference". In: *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30055–30062.

[2] James Reuther et al. "Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation". In: *34th aerospace sciences meeting and exhibit*. 1996, p. 94.

[3] Martina Hasenjäger et al. "Three dimensional evolutionary aerodynamic design optimization with CMA-ES". In: *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. 2005, pp. 2173–2180.

[4] Mohamed Jebalia et al. "Identification of the isotherm function in chromatography using CMA-ES". In: *2007 IEEE Congress on Evolutionary Computation*. IEEE. 2007, pp. 4289–4296.

[5] Atul Agrawal and Phaedon-Stelios Koutsourelakis. "A probabilistic, data-driven closure model for RANS simulations with aleatoric, model uncertainty". In: *Journal of Computational Physics* (2024), p. 112982. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2024.112982. URL: https://www.sciencedirect.com/science/article/pii/S0021999124002316.

[6] Atul Agrawal et al. "From concrete mixture to structural design–a holistic optimization procedure in the presence of uncertainties". In: *arXiv preprint arXiv:2312.03607* (2023).

[7] Maximilian Rixner and Phaedon-Stelios Koutsourelakis. "Self-supervised optimization of random material microstructures in the small-data regime". In: *npj Computational Materials* 8.1 (2022), p. 46.

[8] Tommaso Dorigo et al. "Toward the end-to-end optimization of particle physics instruments with differentiable programming". In: *Reviews in Physics* 10 (June 2023), p. 100085. ISSN: 2405-4283. DOI: 10.1016/j.revip.2023.100085. URL: https://www.sciencedirect.com/science/article/pii/S2405428323000047 (visited on 05/21/2024).

[9] Jared J Thomas et al. "A comparison of eight optimization methods applied to a wind farm layout optimization problem". In: *Wind Energy Science Discussions* 2022 (2022), pp. 1–43.

[10] Sergey Shirobokov et al. "Black-box optimization with local generative surrogates". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 14650–14662.

[11] Zhe Zhang, Minghao Song, and Xiaobiao Huang. "Online accelerator optimization with a machine learning-based stochastic algorithm". In: *Machine Learning: Science and Technology* 2.1 (Dec. 2020), p. 015014. DOI: 10.1088/2632-2153/abc81e. URL: https://dx.doi.org/10.1088/2632-2153/abc81e.

[12] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3 (1992), pp. 229–256.

[13] Nataniel Ruiz, Samuel Schulter, and Manmohan Chandraker. "Learning to simulate". In: *arXiv preprint arXiv:1810.02513* (2018).

[14] German Ros et al. "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3234–3243.

[15] Joaquim RRA Martins and Andrew Ning. *Engineering design optimization*. Cambridge University Press, 2021.

[16] Mathieu Blondel and Vincent Roulet. "The Elements of Differentiable Programming". In: *arXiv preprint arXiv:2403.14606* (2024).

[17] Jonas Degrave, Michiel Hermans, Joni Dambre, et al. "A differentiable physics engine for deep learning in robotics". In: *Frontiers in neurorobotics* (2019), p. 6.

[18] Filipe de Avila Belbute-Peres et al. "End-to-end differentiable physics for learning and control". In: *Advances in neural information processing systems* 31 (2018).

[19] Didier Lucor, Atul Agrawal, and Anne Sergent. "Simple computational strategies for more effective physics-informed neural networks modeling of turbulent natural convection". In: *Journal of Computational Physics* 456 (2022), p. 111022.

[20] Daniel Golovin et al. "Google vizier: A service for black-box optimization". In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 2017, pp. 1487–1495.

[21] Michael B Giles and Niles A Pierce. "An introduction to the adjoint approach to design". In: *Flow, turbulence and combustion* 65.3 (2000), pp. 393–415.

[22] Alexander Luce et al. "Merging automatic differentiation and the adjoint method for photonic inverse design". In: *Machine Learning: Science and Technology* 5.2 (June 2024), p. 025076. DOI: 10.1088/2632-2153/ad5411. URL: https://dx.doi.org/10.1088/2632-2153/ad5411.

[23] Philipp Holl and Nils Thuerey. "Phi-ML: Intuitive Scientific Computing with Dimension Types for Jax, PyTorch, TensorFlow & NumPy". In: *Journal of Open Source Software* 9.95 (2024), p. 6171. DOI: 10.21105/joss.06171. URL: https://doi.org/10.21105/joss.06171.

[24] Sean Gasiorowski et al. "Differentiable simulation of a liquid argon time projection chamber". In: *Machine Learning: Science and Technology* 5.2 (Apr. 2024), p. 025012. DOI: 10.1088/2632-2153/ad2cf0. URL: https://dx.doi.org/10.1088/2632-2153/ad2cf0.

[25] Nathan Baker et al. *Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence*. Tech. rep. USDOE Office of Science (SC), Washington, DC (United States), 2019.

[26] William Moses and Valentin Churavy. "Instead of rewriting foreign code for machine learning, automatically synthesize fast gradients". In: *Advances in neural information processing systems* 33 (2020), pp. 12472–12485.

[27] Jeffrey Larson, Matt Menickelly, and Stefan M Wild. "Derivative-free optimization methods". In: *Acta Numerica* 28 (2019), pp. 287–404.

[28] Charles Audet and Michael Kokkolaras. *Blackbox and derivative-free optimization: theory, algorithms and applications*. 2016.

[29] Jorge J Moré and Stefan M Wild. "Benchmarking derivative-free optimization algorithms". In: *SIAM Journal on Optimization* 20.1 (2009), pp. 172–191.

[30] Rajesh Ranganath, Sean Gerrish, and David Blei. "Black box variational inference". In: *Artificial intelligence and statistics*. PMLR. 2014, pp. 814–822.

[31] Wolfgang Banzhaf et al. *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications*. Morgan Kaufmann Publishers Inc., 1998.

[32] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical bayesian optimization of machine learning algorithms". In: *Advances in neural information processing systems* 25 (2012).

[33] Friedrich Menhorn et al. "A trust-region method for derivative-free nonlinear constrained stochastic optimization". In: *arXiv preprint arXiv:1703.04156* (2017).

[34] Daan Wierstra et al. "Natural evolution strategies". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 949–980.

[35] Krzysztof M Choromanski et al. "From complexity to simplicity: Adaptive es-active subspaces for blackbox optimization". In: *Advances in Neural Information Processing Systems* 32 (2019).

[36] Abhinav Anand, Matthias Degroote, and Alán Aspuru-Guzik. "Natural evolutionary strategies for variational quantum computation". In: *Machine Learning: Science and Technology* 2.4 (July 2021), p. 045012. DOI: 10.1088/2632-2153/abf3ac. URL: https://dx.doi.org/10.1088/2632-2153/abf3ac.

[37] Shakir Mohamed et al. "Monte carlo gradient estimation in machine learning". In: *The Journal of Machine Learning Research* 21.1 (2020), pp. 5183–5244.

[38] Georg Ch Pflug. *Optimization of stochastic models: the interface between simulation and optimization*. Vol. 373. Springer Science & Business Media, 2012.

[39] Gilles Louppe, Joeri Hermans, and Kyle Cranmer. "Adversarial Variational Optimization of Non-Differentiable Simulators". en. In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. ISSN: 2640-3498. PMLR, Apr. 2019, pp. 1438–1447. URL: https://proceedings.mlr.press/v89/louppe19a.html (visited on 11/22/2022).

[40] Atul Agrawal et al. "Multi-fidelity Constrained Optimization for Stochastic Black Box Simulators". In: *arXiv preprint arXiv:2311.15137* (2023).

[41] Arkadij Semenovič Nemirovskij and David Borisovich Yudin. "Problem complexity and method efficiency in optimization". In: (1983).

[42] Anthony Nguyen and Krishnakumar Balasubramanian. "Stochastic zeroth-order functional constrained optimization: Oracle complexity and applications". In: *INFORMS Journal on Optimization* 5.3 (2023), pp. 256–272.

[43] Thomas Bird, Julius Kunze, and David Barber. *Stochastic Variational Optimization*. arXiv:1809.04855 [cs, stat]. Sept. 2018. URL: http://arxiv.org/abs/1809.04855 (visited on 11/08/2022).

[44] Joe Staines and David Barber. *Variational Optimization*. arXiv:1212.4507 [cs, stat]. Dec. 2012. URL: http://arxiv.org/abs/1212.4507 (visited on 11/08/2022).

[45] Jacob R Gardner et al. "Bayesian optimization with inequality constraints." In: *ICML*. Vol. 2014. 2014, pp. 937–945.

[46] Michael JD Powell. *A direct search optimization method that models the objective and constraint functions by linear interpolation*. Springer, 1994.

[47] Dieter Kraft. "A software package for sequential quadratic programming". In: *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt* (1988).

[48] Laurens Bliek et al. "EXPObench: benchmarking surrogate-based optimisation algorithms on expensive black-box functions". In: *arXiv preprint arXiv:2106.04618* (2021).

[49] Aharon Ben-Tal and Arkadi Nemirovski. "Robust solutions of uncertain linear programs". In: *Operations research letters* 25.1 (1999), pp. 1–13.

[50] Dimitris Bertsimas, David B Brown, and Constantine Caramanis. "Theory and applications of robust optimization". In: *SIAM review* 53.3 (2011), pp. 464–501.

[51] I.-J. Wang and J.C. Spall. "Stochastic optimization with inequality constraints using simultaneous perturbations and penalty functions". en. In: *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*. Maui, HI, USA: IEEE, 2003, pp. 3808–3813. ISBN: 978-0-7803-7924-4. DOI: 10.1109/CDC.2003.1271742. URL: http://ieeexplore.ieee.org/document/1271742/ (visited on 10/28/2022).

[52] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[53] Anthony V Fiacco and Garth P McCormick. *Nonlinear programming: sequential unconstrained minimization techniques*. SIAM, 1990.

[54] Giampaolo Liuzzi, Stefano Lucidi, and Marco Sciandrone. "Sequential penalty derivative-free methods for nonlinear constrained optimization". In: *SIAM Journal on Optimization* 20.5 (2010), pp. 2614–2635.

[55] Michel Fortin and Roland Glowinski. *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems*. Elsevier, 2000.

[56] Mark A Beaumont, Wenyang Zhang, and David J Balding. "Approximate Bayesian computation in population genetics". In: *Genetics* 162.4 (2002), pp. 2025–2035.

[57] Paul Marjoram et al. "Markov chain Monte Carlo without likelihoods". In: *Proceedings of the National Academy of Sciences* 100.26 (2003), pp. 15324–15328.

[58] Joe Staines and David Barber. "Optimization by Variational Bounding." In: *ESANN*. 2013.

[59] Peter W Glynn. "Likelihood ratio gradient estimation for stochastic systems". In: *Communications of the ACM* 33.10 (1990), pp. 75–84.

[60] Tim Salimans et al. "Evolution strategies as a scalable alternative to reinforcement learning". In: *arXiv preprint arXiv:1703.03864* (2017).

[61] Frank Sehnke et al. "Parameter-exploring policy gradients". In: *Neural Networks* 23.4 (2010), pp. 551–559.

[62] Yurii Nesterov and Vladimir Spokoiny. "Random gradient-free minimization of convex functions". In: *Foundations of Computational Mathematics* 17.2 (2017), pp. 527–566.

[63] Wouter Kool, Herke van Hoof, and Max Welling. "Buy 4 REINFORCE Samples, Get a Baseline for Free!" en. In: (July 2022). URL: https://openreview.net/forum?id=r1lgTGL5DE (visited on 11/11/2022).

[64] Josef Dick, Frances Y Kuo, and Ian H Sloan. "High-dimensional integration: the quasi-Monte Carlo way". In: *Acta Numerica* 22 (2013), pp. 133–288.

[65] Mark Rowland et al. "Geometrically coupled monte carlo sampling". In: *Advances in Neural Information Processing Systems* 31 (2018).

[66] Paul Glasserman. *Monte Carlo methods in financial engineering*. Vol. 53. Springer, 2004.

[67] Il'ya Meerovich Sobol'. "On the distribution of points in a cube and the approximate evaluation of integrals". In: *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki* 7.4 (1967), pp. 784–802.

[68] Shun-Ichi Amari. "Natural gradient works efficiently in learning". In: *Neural computation* 10.2 (1998), pp. 251–276.

[69] Da Tang and Rajesh Ranganath. "The Variational Predictive Natural Gradient". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 6145–6154. URL: https://proceedings.mlr.press/v97/tang19c.html.

[70] Solomon Kullback and Richard A Leibler. "On information and sufficiency". In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[71] C Radhakrishna Rao. "Information and the accuracy attainable in the estimation of statistical parameters". In: *Breakthroughs in Statistics: Foundations and basic theory*. Springer, 1992, pp. 235–247.

[72] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[73] Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. "Survey of multifidelity methods in uncertainty propagation, inference, and optimization". In: *Siam Review* 60.3 (2018), pp. 550–591.

[74] Michael B Giles. "Multilevel monte carlo methods". In: *Acta numerica* 24 (2015), pp. 259–328.

[75] Stefan Heinrich. "Multilevel monte carlo methods". In: *Large-Scale Scientific Computing: Third International Conference, LSSC 2001 Sozopol, Bulgaria, June 6–10, 2001 Revised Papers 3*. Springer. 2001, pp. 58–67.

[76] Friedrich Menhorn et al. "Multilevel Monte Carlo estimators for derivative-free optimization under uncertainty". In: *International Journal for Uncertainty Quantification* 14.3 (2024).

[77] Adam Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[78] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[79] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[80] Fernando Nogueira. *Bayesian Optimization: Open source constrained global optimization tool for Python*. 2014–. URL: https://github.com/bayesian-optimization/BayesianOptimization.

[81] Victor Picheny, Tobias Wagner, and David Ginsbourger. "A benchmark of kriging-based infill criteria for noisy optimization". In: *Structural and multidisciplinary optimization* 48 (2013), pp. 607–626.

[82] JS Gray et al. "Automatic evaluation of multidisciplinary derivatives using a graph-based problem formulation in OpenMDAO, 15th AIAA". In: *ISSMO Multidisciplinary Analysis and Optimization Conference, Atlanta, Georgia, USA*. 2014.

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

[84] Andrew PJ Stanley and Andrew Ning. "Massive simplification of the wind farm layout optimization problem". In: *Wind Energy Science* 4.4 (2019), pp. 663–676.

[85] Andrés Santiago Padrón et al. "Polynomial chaos to efficiently compute the annual energy production in wind farm layout optimization". In: *Wind Energy Science* 4.2 (2019), pp. 211–231.

[86] Nicholas MK Poon and Joaquim RRA Martins. "An adaptive approach to constraint aggregation using adjoint sensitivity analysis". In: *Structural and Multidisciplinary Optimization* 34 (2007), pp. 61–73.

[87] Jason Jonkman et al. *Definition of a 5-MW reference wind turbine for offshore system development*. Tech. rep. National Renewable Energy Lab.(NREL), Golden, CO (United States), 2009.

[88] NREL. *FLORIS. Version 1.0.0*. 2019. URL: https://github.com/NREL/floris.

[89] Niels Otto Jensen. *A note on wind generator interaction*. Risø National Laboratory, 1983.

[90] Jennifer King et al. "Control-oriented model for secondary effects of wake steering". In: *Wind Energy Science* 6.3 (2021), pp. 701–714.

[91] John Jasa et al. "Effectively using multifidelity optimization for wind turbine design". In: *Wind Energy Science Discussions* 2021 (2021), pp. 1–22.

[92] Anuj Karpatne, Ramakrishnan Kannan, and Vipin Kumar. *Knowledge Guided Machine Learning: Accelerating Discovery Using Scientific Knowledge and Data*. CRC Press, 2022.