Enabling Long-range Exploration in Minimization of Multimodal Functions (Supplementary Material)

Jiaxin Zhang[1] **Hoang Tran**[1] **Dan Lu**[2] **Guannan Zhang**[1]

[1]Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA
[2]Computational Sciences and Engineering Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

1 ADDITIONAL INFORMATION ON THE HIGH-DIMENSIONAL BENCHMARK FUNCTION TESTS

We provide definitions of the high-dimensional benchmark functions and the implementation details of the DGS method and the other baselines.

1.1 DEFINITIONS OF THE BENCHMARK FUNCTIONS

To make the test functions more general, we applied the following linear transformation to $\boldsymbol{x}$, i.e.,

$$\boldsymbol{z} = \mathbf{R}(\boldsymbol{x} - \boldsymbol{x}_{\text{opt}}),$$

where $\mathbf{R}$ is a rotation matrix making the functions non-separable and $\boldsymbol{x}_{\text{opt}}$ is the optimal state. Then we substitute $\boldsymbol{z}$ into the standard definitions of the benchmark functions to formulate our test problems. For notational simplicity, we use $\boldsymbol{z}$ as the input variable in the following definitions and omit the dependence of $\boldsymbol{z}$ on $\boldsymbol{x}$.

- The **Ellipsoidal** function $F_1(\boldsymbol{x})$ is defined by

$$F_3(\boldsymbol{x}) = \sum_{i=1}^{d} 10^{6\frac{i-1}{d-1}} z_i^2,$$

where d is the dimension and $\boldsymbol{x} \in [-2, 2]^d$ is the input domain. The global minimum is $f(\boldsymbol{x}_{\text{opt}}) = 0$. This represents *convex and highly ill-conditioned* landscapes.

- The **Sharp Ridge** function $F_2(\boldsymbol{x})$ is defined by

$$F_2(\boldsymbol{x}) = z_1^2 + 100\sqrt{\sum_{i=2}^{d} z_i^2},$$

where d is the dimension and $\boldsymbol{x} \in [-10, 10]^d$ is the input domain. The global minimum is $f(\boldsymbol{x}_{\text{opt}}) = 0$. This represents *convex and anisotropic* landscapes. There is a sharp ridge defined along $z_2^2 + \cdots + z_d^2 = 0$ that must be followed to reach the global minimum, which creates difficulties for optimizations algorithms.

- The **Ackley** function $F_3(\boldsymbol{x})$ is defined by

$$F_3(\boldsymbol{x}) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} z_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cz_i)\right) + a + \exp(1),$$

where d is the dimension and $a = 20, b = 0.2, c = 2\pi$ are used in our experiments. The input domain $\boldsymbol{x} \in [-32.768, 32.768]$. The global minimum is $f(\boldsymbol{x}_{\text{opt}}) = 0$ The Ackley function represents *non-convex* landscapes with *nearly flat outer region*. The function poses a risk for optimization algorithms, particularly hill-climbing algorithms, to be trapped in one of its many local minima.

- The **Rastrigin** function $F_4(\boldsymbol{x})$ is defined by

$$F_4(\boldsymbol{x}) = 10d + \sum_{i=1}^{d}[z_i^2 - 10\cos(2\pi z_i)], \tag{1}$$

where d is the dimension and $\boldsymbol{x} \in [-5.12, 5.12]^d$ is the input domain. The global minimum is $f(\boldsymbol{x}_{\text{opt}}) = 0$. This function represents *multimodal and separable* landscapes.

- The **Schaffer** function $F_5(\boldsymbol{x})$ is defined by

$$F_5(\boldsymbol{x}) = \frac{1}{d-1}\left(\sum_{i=1}^{d-1}\left(\sqrt{s_i} + \sqrt{s_i}\sin^2(50 s_i^{\frac{1}{5}})\right)\right)^2$$
$$\text{with } s_i = \sqrt{z_i^2 + z_{i+1}^2},$$

where $\boldsymbol{x} \in [-100, 100]^d$ is the input domain. The global minimum is $f(\boldsymbol{x}_{\text{opt}}) = 0$. This function represents *multimodal and non-separable* landscapes.

- The **Schwefel** function $F_6(\boldsymbol{x})$ is defined by

$$F_6(\boldsymbol{x}) = 418.9829d - \sum_{i=1}^{d} z_i \sin(\sqrt{|z_i|}),$$

where $\boldsymbol{x} \in [-500, 500]^d$ is the input domain. The global minimum is $f(\boldsymbol{x}_{\text{opt}}) = 0$, at $\boldsymbol{z} = (420.9687, \cdots, 420.9687)$. This function represents *multimodal* landscapes with *no global structure*.

1.2 EXPERIMENTAL DETAILS

1.2.1 The DGS method

There are six hyperparameters given in Algorithm 1, i.e., M: the number of GH quadrature points; λ_t: learning rate; α: the scaling factor for the rotation $\Delta\Xi$; r, β: the mean and variation for sampling σ; γ: the tolerance for triggering random perturbation. We turned off the random perturbation by setting γ to a very small number, such that we don't need to worry about $\Delta\Xi$, r and β. The rest of the hyperparameters are set as follows.

The hyperparameters of the DGS method are fixed for the 6 test functions. Specifically, we used $M = 5$ GH quadrature points. A quadratic decay schedule

$$\lambda_t = (\lambda_0 - \lambda_T)\left(1 - \frac{t}{T}\right)^2 + \lambda_T,$$

is used for the learning rate where the maximum number of iterations is set to $T = 200$. The initial (maximum) learning rate λ_0 is set to 5% of the diagonal length of the d-dimensional search domain, and the terminal (minimum) learning rate λ_T is set to 1% of λ_0. A quadratic decay schedule

$$\sigma_t = (\sigma_0 - \sigma_T)\left(1 - \frac{t}{T}\right)^2 + \sigma_T,$$

is used for the smoothing radius where the maximum number of iterations is set to $T = 200$. The initial smoothing radius σ_0 is 5 times of the length of the search domain for each variable, and the terminal smoothing radius σ_T is 1% of the initial radius σ_0.

1.2.2 The ES-Bpop method

ES-Bpop refers to the standard OpenAI evolution strategy in [Salimans et al., 2017] with the a big population, i.e., the same population size as the DGS method. The purpose of using a big population is to compare the MC-based estimator for the standard GS gradient and the DGS gradient given the same computational cost. In this setting, the ES-Bpop only has two hyperparameters, i.e., the learning rate λ_t and the smoothing radius σ_t. We use the same decay schedules for λ and σ as the DGS method, so the only difference between DGS and ES-Bpop is that DGS uses the nonlocal gradient and ES-Bpop uses the local gradient.

1.2.3 The ASEBO method

ASEBO refers to Adaptive ES-Active Subspaces for Blackbox Optimization proposed in [Choromanski et al., 2019]. This is the state-of-the-art method in the family of ES. It has been shown that other recent developments on ES, e.g., [Akimoto and Hansen, 2016, Loshchilov et al., 2019], underperform ASEBO in optimizing the benchmark functions. We use the code published at `https://github.com/jparkerholder/ASEBO` by the authors of the ASEBO method. Since we use ASEBO to represent the state-of-the-art ES method, we set the population size to the standard value $4 + 3\log(d)$. Due to the use of a small population, the smoothing radius σ needs to be small to control the variance of the MC estimator. It turns out that $\sigma = 0.1$ works the best for the test cases. The ASEBO code uses $\lambda_t = \alpha\lambda_{t-1}$ to set a learning rate decay schedule. For the six test cases, we tune α by searching the grid $\{0.999, 0.99, 0.9\}$ and choose $\alpha = 0.99$ for all the six functions. Again, due to the small population size, the initial learning rate λ_0 cannot be set to as large as those for DGS or ES-Bpop because it will overshoot. Thus, we search for the initial learning rate on the grid $\{0.01, 0.1, 0.5, 1.0\}$, and it turns out $\lambda_0 = 0.1$ provides the best performance for ASEBO for the test functions.

1.2.4 The IPop-CMA method

IPop-CMA refers to the restart covariance matrix adaptation evolution strategy with increased population size proposed in [Auger and Hansen, 2005]. We use the code pycma v3.0.3 available at `https://github.com/CMA-ES/pycma`. The main subroutine we use is `cma.fmin`, in which the hyperparameters are

- `restarts=9`: the maximum number of restarts with increasing population size;
- `restart_from_best=False`: which point to restart from;
- `incpopsize=2`: multiplier for increasing the population size before each restart;
- σ_0: the initial exploration radius is set to 1/4 of the search domain width.

1.2.5 The Nesterov method

Nesterov refers to the random search method proposed in [Nesterov and Spokoiny, 2017]. We use the stochastic oracle

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \lambda_t F'(\boldsymbol{x}_t, \boldsymbol{u}_t),$$

where $\boldsymbol{u}_t$ is a randomly selected direction and $F'(\boldsymbol{x}_t, \boldsymbol{u}_t)$ is the directional derivative along $\boldsymbol{u}_t$. According to the analysis in [Nesterov and Spokoiny, 2017], this oracle is more powerful and can be used for non-convex non-smooth functions. As suggested in [Nesterov and Spokoiny, 2017], we use forward difference scheme to compute the directional derivative. The only hyperparameter is the learning rate λ_t. We use the same quadratic decay model as in DGS and the tuned learning rate schedule is given in Table 1

Table 1: The hyperparameter values for Nesterov

	λ_0	λ_T	T
Ellipsoidal 2000D	0.001	0.0001	1,000,000
Sharp-Ridge 2000D	0.001	0.0001	1,000,000
Ackley 2000D	1.0	0.001	1,000,000
Rastrigin 2000D	0.00001	0.000001	1,000,000
Schaffer 2000D	0.0001	0.0001	1,000,000
Schwefel 2000D	0.005	0.0001	1,000,000
Sphere 20D	0.01	0.001	800
Sharp-Ridge 20D	0.001	0.00001	2000
Ackley 20D	0.1	0.01	8000
Rastrigin 20D	0.001	0.0001	6000
Schaffer 20D	0.005	0.001	8000
Schwefel 20D	0.1	0.01	10000

1.2.6 The FD method

FD refers to the classical central difference scheme for local gradient estimation. The only hyperparameter is the learning rate λ_t. We use the same quadratic decay model as in DGS and the tuned learning rate schedule is given in Table 2.

Table 2: The hyperparameter values for FD

	λ_0	λ_T	T
Ellipsoidal 2000D	1.0	0.01	20
Sharp-Ridge 2000D	0.4	0.0001	60
Ackley 2000D	1.0	0.001	160
Rastrigin 2000D	0.001	0.0001	400
Schaffer 2000D	0.01	0.001	600
Schwefel 2000D	0.1	0.01	500
Sphere 20D	1.0	0.01	40
Sharp-Ridge 20D	0.4	0.0001	50
Ackley 20D	1.0	0.01	400
Rastrigin 20D	0.01	0.001	300
Schaffer 20D	0.001	0.0001	400
Schwefel 20D	0.1	0.01	500

1.2.7 The Cobyla method

Cobyla refers to Constrained Optimization By Linear Approximation optimizer [Powell, 1994]. This algorithm is based on linear approximations to the objective function and each constraint, and can be used for constrained problems where the derivative of the objective function is not known. We use `scipy.optimize.minimize` with default hyperparameters for COBYLA implementation.

1.2.8 The Powell method

Powell is a conjugate direction method without calculating derivatives [Powell, 1964]. It performs sequential one-dimensional minimizations along each vector of the directions set, which is updated at each iteration of the main minimization loop. The function need not be differentiable, and no derivatives are taken. We implement the Powell method using `scipy.optimize.minimize` with default hyperparameter setting in `scipy`.

1.2.9 The DE method

DE refers to Differential Evolution method [Storn and Price, 1997] that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. DE method is stochastic in nature to find the minimum, and can search large areas of candidate space, but often requires larger numbers of function evaluations than conventional gradient-based techniques. We use the `scipy.optimize.differential_evolution` implementation and default hyperparameters.

1.2.10 The PSO method

PSO refers to Particle Swarm Optimization method [Kennedy and Eberhart, 1995], which solves a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. PSO is one of classical evolutionary algorithm, which means it does not require that the optimization problem be differentiable. We use the code `PYSWARMS` [Miranda, 2018] available at `https://github.com/ljvmiranda921/pyswarms`. The hyperparameters are

- `n_particle`=100: the number of particle
- `c1=0.5,c2=0.3,w=0.9`: PYSWARMS option parameters

1.2.11 The TuRBO method

TuRBO refers to Trust Region Bayesian optimization [Eriksson et al., 2019] which is the state-of-the-art Bayesian optimization method. We use the code published at `https://github.com/uber-research/TuRBO` by the authors. The hyperparameters of TuRBO are listed as follows:

- `n_init=10`: number of initial bounds from an symmetric Latin hypercube design

- `n_trust_regions=5`: number of trust regions
- `batch_size=10`: how large batch size TuRBO uses
- `max_cholesky_size=2000`: switch from Cholesky to Lanczos
- `n_training_steps=50`: number of steps of ADAM to learn the hyperparameters

1.3 ADDITIONAL EXPERIMENTS ON LOW-DIMENSIONAL CASES

Besides high-dimensional problems, it is also natural to compare the performance of the DGS method with the other baselines in solving relatively low-dimensional problems. To this end, we conduct another set of tests for the functions in 20-dimensional spaces. The hyperparameters used for the 20D tests are also given in Section 1. The main results are given in Figure 1. We have the following observations by comparing the 2000D and 20D results and hyperparameter values used for DGS and the baselines.

The DGS method can use more aggressive learning rate schedules to accelerate convergence. This is due to the good smoothing effect of the DGS gradient and the high accuracy of the DGS estimator. For example, the Ackley function has a very large and flat outer-region, and the convergence speed depends on how fast an optimizer can go through the flat region and get to the mode containing the global minimum. DGS can use a very large initial learning rate e.g., $\lambda_0 = 8000$ for the 2000D case, without worry about overshooting, because its cosine distance is small and the variance of the DGS estimator is also small. In comparison, ES-Bpop, Nesterov and FD need to use much smaller learning rates to avoid overshooting; and ES-Bpop performs better than Nesterov and FD due to the use of a relatively large population. Another reason why Nesterov and FD need to use smaller learning rates is that the local derivatives/gradients of some test functions are very fluctuating, e.g., Schaffer and Rastrigin, and both methods do not provide sufficient smoothing effect to reduce the fluctuation.

The advantage of the DGS in the 20D case is not as significant as in the 2000D case. For example, IPop-CMA outperforms DGS in minimizing the 20D Ackley function. The convergence speed of IPop-CMA type methods depends on how fast samples can be dropped in the mode containing the global minimum. The flat outer-region of 20D Ackley is much smaller than that of the 2000D Ackley, so that it is easier for IPop-CMA to have a sample dropped in the mode containing the global minimum. For the Sharp Ridge function, FD outperforms DGS because the sharp ridge defined by $x_2^2 + \cdots + x_d^2 = 0$ is easier to follow in the 20D space. The landscapes without any global structures, e.g., the Schwefel function, is still difficult to minimize in 20D cases.

2 ADDITIONAL INFORMATION ON THE CONSTRAINED TOPOLOGY OPTIMIZATION

2.1 TOPOLOGY OPTIMIZATION MATHEMATICAL FORMULATION

Here we provide more background information about the topology optimization (TO) problem tested in §4.2. TO is a mathematical method that aims to optimize material layout defined on a design domain Ω with given boundary conditions, loads and volume constraint, to minimize structural compliance C, or equivalently, the least strain energy. In this work, we use the modified Solid Isotropic Material with Penalization (SIMP) approach [Sigmund, 2007] with design-based approach to topology optimization, where each element e is assigned a density x_e that determines its Young's modulus E_e:

$$E_e(x_e) = E_{\min} + x_e^p(E_0 - E_{\min}), \quad x_e \in [0,1] \tag{2}$$

where E_0 is the stiffness of the material, $E_{\min}$ is a very small stiffness assigned to void regions to prevent the stiffness matrix becoming singular. The modified SIMP approach differs from the classical SIMP approach [Bendsøe, 1989], where elements with zero stiffness are avoided by using a small value. The modified mathematical formulation of the considered TO problem is

$$\begin{aligned} \min_{\boldsymbol{x}} : \quad & C(\boldsymbol{x}) = \mathbf{U}^T\mathbf{K}\mathbf{U} = \sum_{e=1}^{N} E_e(x_e)\mathbf{u}_e^T\mathbf{k}_0\mathbf{u}_e \\ s.t. : \quad & V(\boldsymbol{x})/V_0 \le \zeta \\ : \quad & \mathbf{K}\mathbf{U} = \mathbf{F} \\ : \quad & 0 \le \boldsymbol{x} \le 1 \end{aligned} \tag{3}$$

where $\boldsymbol{x}$ is the vector of design variables, C is the structural compliance, $\mathbf{K}$ is the global stiffness matrix, $\mathbf{U}$ and $\mathbf{F}$ are the global displacement and force vectors respectively, $\mathbf{u}_e$ and $\mathbf{k}_e$ are the element displacement vector and stiffness matrix respectively, N is the number of elements used to discretize the design domain Ω, $V(\boldsymbol{x})$ and V_0 are the material volume and design domain volume respectively, ζ is the prescribed volume fraction, and p is the penalization power coefficient (typically $p = 3$).

2.2 THE TOPOLOGY OPTIMIZATION PROBLEM

The experimental TO example in Section 4.2 is a typical structural bridge design problem. Our goal is to minimize the structural compliance subject to unit uniform pressure on the top of the design domain Ω, and two fixed supports on the bottom of the design domain Ω, as shown in Fig.2. The volume fraction is $\zeta = 0.2$. The design domain Ω is discretized by 120×40 elements. Using symmetric boundary

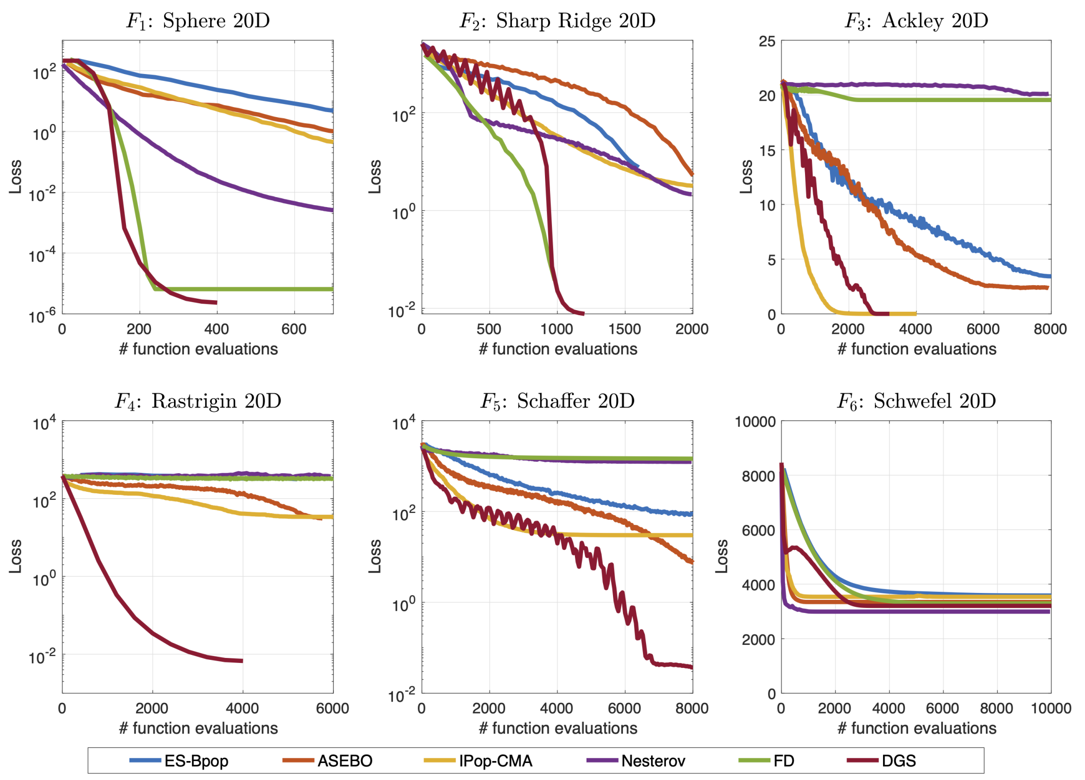

Figure 1: Comparison of the loss decay w.r.t. # function evaluations for the 6 benchmark functions in 20-dimensional spaces. Each curve was generated by averaging 20 independent trials with random initialization. The global minimum is $F(\boldsymbol{x}) = 0$ for all the six functions.

condition, we reduce the whole design domain into an half to save computational cost. As a result, there are total 2400 (60×40) design variables.

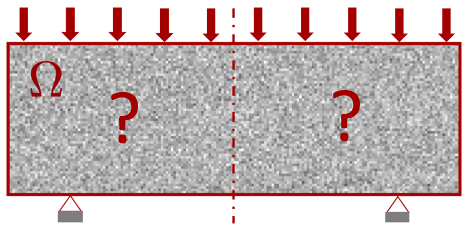

Figure 2: Illustration of the structural bridge design

There has been several mathematical challenges in solving TO problem, as shown in Eq.(3), which can be briefly summarized as follows:

- *Rigid constrained optimization*. Typically, design criteria are specified by user to satisfy multiple constraints including material volume fraction, maximum stress and geometry constraints, so that meaningful structures can be obtained.

- *High dimensional design space*. This is because each element is identified as an independent design variable in TO, shown in Eq.(3). In most cases, it requires a large number of elements to ensure the accuracy of finite element method (FEM) and to perform a clear final topology.

- *Highly nonconvex propriety*. TO introduces the SIMP approach to convert the 0-1 integer optimization into continuous optimization but it also changes TO to a difficult multi-modal optimization problem where a large number of local minima exist.

- *Optimum depends on the initial design*. TO has challenges to pursue global minima [Bendsoe and Sigmund, 2013] due to a limited capability of global exploration in sensitivity analysis when using adjoint methods or finite difference methods. The different initial guesses therefore finally lead to different local minima.

These challenges make a few optimization methods infeasible in solving TO problem well. Bayesian optimization (BO) is good at pursuing global minima but has limitations in high-dimensional problems ($D > 1000$). While many improvements have been proposed Wu et al. [2017], Eriksson et al. [2018] to mitigate this challenge, the constraints in BO framework is still a critical issue [Gardner et al., 2014], specifically in practical implementation. The widely used training algorithms including SGD, Adam, RMSprop, etc., are inapplicable to TO problems without handling constraints.

2.3 THE CONSTRAINED OPTIMIZATION FOR TOPOLOGY OPTIMIZATION

Method of Moving Asymptotes (MMA) [Svanberg, 1987] algorithm is the state-of-the-art optimizer, which has been demonstrated to be versatile and well suited for wide range TO problems. The basic of MMA aims at solving general nonlinear constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{x}}: \quad & f_0(\mathbf{x}) + a_0 z + \sum_{i=1}^{m}(c_i y_i + \frac{1}{2} d_i y_i^2) \\ s.t.: \quad & f_i(\mathbf{x}) - a_i z - y_i \le 0, \quad i = 1, ..., m \\ : \quad & \mathbf{x} \in X, \mathbf{y} \ge 0, z \ge 0 \end{aligned} \tag{4}$$

Here, $X = \{x \in \mathbb{R}^n | x_j^{\min} \le x_j \le x_j^{\max}, j = 1, ..., n\}$, where $x_j^{\min}$ and $x_j^{\max}$ are given real numbers which satisfy $x_j^{\min} < x_j^{\max}$ for all j, $f_0, f_1, ..., f_m$ are given, continuously differentiable, real-valued functions on X, a_0, a_i, c_i and d_i are given real numbers which satisfy $a_0 > 0, a_i \ge 0, c_i \ge 0$ and $d_i \ge 0$ and $c_i + d_i > 0$ for all i and also $a_i c_i > a_0$ for all i with $a_i > 0$.

MMA is a gradient-based method for solving Eq.(4) using the following steps. In each iteration, given the current point $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}, z^{(k)})$, MMA generates an approximating subproblem, where the functions $f_i(\mathbf{x})$ are replaced by convex functions $\hat{f}_i^{(k)}(\mathbf{x})$. The approximating functions are determined by the *gradient information* at the current iteration point and moving asymptotes parameters which are updated in each iteration based on information from previous iteration points. The next iteration point $(\mathbf{x}^{(k+1)}, \mathbf{y}^{(k+1)}, z^{(k+1)})$ is obtained by solving the subproblem, which looks as follows:

$$\begin{aligned} \min_{\mathbf{x}}: \quad & \hat{f}_0^{(k)}(\mathbf{x}) + a_0 z + \sum_{i=1}^{m}(c_i y_i + \frac{1}{2} d_i y_i^2) \\ s.t.: \quad & \hat{f}_i^{(k)}(\mathbf{x}) - a_i z - y_i \le 0, \quad i = 1, ..., m \\ : \quad & \alpha_j^{(k)} \le x_j \le \beta_j^{(k)}, \quad j = 1, ..., m \\ : \quad & y_i \ge 0, i = 1, ..., m \\ : \quad & z \ge 0 \end{aligned} \tag{5}$$

where the approximating functions $\hat{f}_i^{(k)}(\mathbf{x})$ are chosen as

$$\hat{f}_i^{(k)}(\mathbf{x}) = \sum_{j=1}^{n} \left(\frac{p_{ij}^{(k)}}{u_j^{(k)} - x_j} + \frac{q_{ij}^{(k)}}{x_j - l_j^{(k)}} \right) + r_i^{(k)}, \tag{6}$$

for $i = 0, 1, ..., m$, where

$$\begin{aligned} p_{ij}^{(k)} &= (u_j^{(k)} - x_j^{(k)})^2 \Bigg(1.001 \left(\frac{\partial f_i}{\partial x_j}(\mathbf{x}^{(k)}) \right)^+ \\ &\quad + 0.001 \left(\frac{\partial f_i}{\partial x_j}(\mathbf{x}^{(k)}) \right)^- + \frac{10^{-5}}{x_j^{\max} - x_j^{\min}} \Bigg), \\ q_{ij}^{(k)} &= (x_j^{(k)} - l_j^{(k)})^2 \Bigg(0.001 \left(\frac{\partial f_i}{\partial x_j}(\mathbf{x}^{(k)}) \right)^+ \\ &\quad + 1.001 \left(\frac{\partial f_i}{\partial x_j}(\mathbf{x}^{(k)}) \right)^- + \frac{10^{-5}}{x_j^{\max} - x_j^{\min}} \Bigg), \\ r_i^{(k)} &= \hat{f}_i(\mathbf{x}^{(k)}) - \sum_{j=1}^{n} \left(\frac{p_{ij}^{(k)}}{u_j^{(k)} - x_j} + \frac{q_{ij}^{(k)}}{x_j - l_j^{(k)}} \right). \end{aligned}$$

Here, $\left(\frac{\partial f_i}{\partial x_j}(\mathbf{x}^{(k)}) \right)^+$ denotes the largest of the two numbers $\frac{\partial f_i}{\partial x_j}(\mathbf{x}^{(k)})$ and 0, while $\left(\frac{\partial f_i}{\partial x_j}(\mathbf{x}^{(k)}) \right)^-$ denotes the largest of the two numbers $-\frac{\partial f_i}{\partial x_j}(\mathbf{x}^{(k)})$ and 0.

The central advantage of MMA in TO is the use of *separable* and *convex* approximations. The separable property means that the necessary conditions of the subproblems do not couple the primary variables and the latter means that dual methods or primal-dual methods can be employed [Bendsoe and Sigmund, 2013]. Combined both two can significantly reduce the computational cost needed to solve the subproblems, particularly for problems with multiple constraints.

2.4 EXPERIMENTAL DETAILS

2.4.1 Implementation of the DGS method in TO

We solve the TO design problem shown in Fig. 2 by inserting the DGS gradient into the MMA optimizer, as discussed in main paper. Our idea is to exploit the nonlocal exploration ability of the DGS gradient to find a better design. The implementation of the DGS algorithm for TO problem can be summarized as the following steps:

- Make an initial design using Gaussian random noise.
- For the given distribution of density, compute the displacement using FEM.
- Compute the objective, typically the compliance of this design, and the associated gradient with respect to design changes using DGS. If the change is smaller than the specific threshold, stop the iteration, otherwise continue.
- Compute the update of the density variable, by solving the MMA approximation subproblem using a dual or primal-dual method.
- Repeat the iteration loop.

Our implementation is built on the python implementation published by [Andreassen et al., 2011] at `http://www.topopt.mek.dtu.dk/Apps-and-software` and we choose density-based filtering with filter radius f_r=1.5 to avoid the numerical instability like checkerboard problem. For MMA optimizer, we use the python implementation from `https://github.com/arjendeetman/TopOpt-MMA-Python` with the default hyperparameters defined in MMA. To reduce the computational cost, we use mpi4py to parallelly run the TO example on multiple cores workstation.

2.4.2 hyperparameters of each method in TO

The hyperparameters of DGS in TO design problem are $M = 5$, $\alpha = 0.1$, $r = 0.25$, $\beta = 0.2$ and $\gamma = 0.01$. Note that there is no learning rate λ in this case because the update step of design variable is achieved by MMA optimizer. The hyperparameters for other compared methods include: (1) ES-Bpop: $\sigma = 0.25$ and $\lambda_t = 0.99\lambda_{t-1}$ with $\lambda_0 = 0.1$; (2) ASEBO: the population size is $14 \approx 4 + 3\log(2400)$, $\sigma = 0.1$ and $\lambda_t = 0.99\lambda_{t-1}$ with $\lambda_0 = 0.1$; (3) IPop-CMA: `restarts=9`, `restart_from_best=False`, `incpopsize = 2`, $\sigma_0 = 0.25$; (4) Nesterov: $\lambda_t = 0.99\lambda_{t-1}$ with $\lambda_0 = 0.01$; (5) FD: $\lambda_t = 0.99\lambda_{t-1}$ with $\lambda_0 = 0.01$; (6) Cobyla: the default setting in `scipy` (7) Powell: the default setting in `scipy`; (8) DE: the default setting in `scipy`; (9) PSO: the number of particles is 100; (10) TuRBO: the number of trust regions is 5, the batch size is 10, the initial sample size is 10.

2.5 ADDITIONAL DISCUSSION

The results demonstrate that the IPop-CMA underperforms all other methods since it fails to solve such constrained optimization well. The main challenge for IPop-CMA is its limitation in efficiently handling constraints because of its sample-based update mechanism, and in effectively incorporating with other optimizer, such as MMA. To satisfy the volume constraint in TO problem, IPop-CMA uses Lagrangian penalty as regularization term to enforce the constraints but loses its own capability to seek optimization. The regularization coefficient is highly sensitive to balance the penalty term and loss function in pycma (v3.0.3 available at `https://github.com/CMA-ES/pycma`) with constraints. In other words, it is difficult for IPop-CMA to address constrained optimization by using a simple penalty approach. This has also been the challenges of CMA-based approaches in complex real-world engineering applications.

3 ADDITIONAL INFORMATION ON THE HYDROLOGY INFERENCE EXAMPLE

We provide detailed information on the implementation of the hydrology example. We consider a 2D square aquifer domain, denoted by $\mathcal{D} = [0,1] \times [0,2]$. The domain the aquifer is discritized into a 100×200 mesh. The partial differential equation (PDE) governing the groundwater flow is

$$\begin{aligned}
-\nabla \cdot (a(\boldsymbol{x})\nabla u(\boldsymbol{x})) &= 0 \quad && \text{for } \boldsymbol{x} \in (0,1) \times (0,2)\\
\nabla u(\boldsymbol{x}) &= 0 && \text{for } x_2 = 0 \text{ and } x_2 = 2\\
u(\boldsymbol{x}) &= 10 && \text{for } x_1 = 0\\
u(\boldsymbol{x}) &= 100 && \text{for } x_1 = 1,
\end{aligned}$$

where $u(\boldsymbol{x})$ is the hydraulic head field and $a(\boldsymbol{x})$ is the hydraulic conductivity field. The no-flow boundary condition $\nabla u = 0$ is imposed to the top $(x_2 = 2)$ and bottom $(x_2 = 0)$ boundaries, and the constant hydraulic head condition is applied to the left $(x_1 = 0)$ and right $(x_1 = 1)$ boundary. The goal is to infer $a(\boldsymbol{x})$ using the sampled data of $u(\boldsymbol{x})$.

The ground-truth of the hydraulic conductivity field (shown in Figure 7 (Left)) is generated by a sequential Gaussian sampling. We use a full connected neural network (FNN) to approximate the logarithm of $a(\boldsymbol{x})$, i.e., FNN$(\boldsymbol{x};\boldsymbol{w}) \approx \log(a(\boldsymbol{x}))$. The loss function is defined by

$$Loss = \frac{1}{S}\sum_{s=1}^{S}\left[\text{MODFLOW}(\text{FNN}(\boldsymbol{x}))(\boldsymbol{x}_s) - u(\boldsymbol{x}_s)\right]^2,$$

where $S = 50$, and $u(\boldsymbol{x}_s)$ for $s = 1, \ldots, S$ are the hydraulic head data sampled at 50 random locations. The simulator MODFLOW$(\cdot)$ maps the predicted hydraulic conductivity field FNN$(\boldsymbol{x})$) to the hydraulic head by solving the above PDE.

The hyperparameters are set as follows. DGS: $M = 5, \alpha = 0.1, r = 0.1, \beta = 0.1, \gamma = 0.001$ and $\lambda_t = 0.99\lambda_{t-1}$ with $\lambda_0 = 0.1$; ES-Bpop: $\sigma = 0.1$ and $\lambda_t = 0.99\lambda_{t-1}$ with $\lambda_0 = 0.1$; ASEBO: $\sigma = 0.1$, $\lambda_t = 0.99\lambda_{t-1}$ with $\lambda_0 = 0.1$; IPop-CMA: `restarts=9`, `restart_from_best=False`, `incpopsize=2`, $\sigma_0 = 0.3$; Nesterov: $\lambda_t = 0.99\lambda_{t-1}$ with $\lambda_0 = 0.05$; FD: $\lambda_t = 0.99\lambda_{t-1}$ with $\lambda_0 = 0.05$; Cobyla: the default hyperparameters in `scipy` Powell: the default hyperparameters in `scipy`; (8) DE: the default hyperparameters in `scipy`; (9) PSO: the number of particles is 100; (10) TuRBO: the number of trust regions is 5, the batch size is 10, the initial sample size is 10.

4 ADDITIONAL DISCUSSION ON THE ASYMPTOTIC CONSISTENCY OF THE DGS GRADIENT

We provide additional results to support the discussion on the asymptotic consistency. Recall that $F \in C^{1,1}(\mathbb{R}^d)$ if there exists $L > 0$ such that $\|\nabla F(\boldsymbol{x}+\boldsymbol{\xi}) - \nabla F(\boldsymbol{x})\| \leq L\|\boldsymbol{\xi}\|$, $\forall \boldsymbol{x}, \boldsymbol{\xi} \in \mathbb{R}^d$. Also, recall the error of one-dimensional GH quadrature

$$\big|(\widetilde{\mathscr{D}}^M - \mathscr{D})[G_\sigma]\big| \leq C\frac{M\,!\sqrt{\pi}}{2^M(2M)\,!}, \tag{7}$$

where $C > 0$ is a constant independent of M and σ. Given a unit vector $\boldsymbol{\xi} \in \mathbb{R}^d$, we define $\nabla_{\boldsymbol{\xi}} F(\boldsymbol{x})$ the partial derivatives of F at $\boldsymbol{x} \in \mathbb{R}^d$ in direction $\boldsymbol{\xi}$. We say F is a strongly convex function if there exists a positive number τ such that for any $\boldsymbol{x}, \boldsymbol{\xi} \in \mathbb{R}^d$, $F(\boldsymbol{x}+\boldsymbol{\xi}) \geq F(\boldsymbol{x}) + \langle \nabla F(\boldsymbol{x}), \boldsymbol{\xi}\rangle + \frac{\tau}{2}\|\boldsymbol{\xi}\|^2$. We call τ the convexity parameter of F. We prove below the estimate on the difference between DGS estimator $\widetilde{\nabla}^M_{\sigma,\boldsymbol{\Xi}}[F]$ and ∇F.

Proposition 1 *Let $\Xi = \{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_d\}$ be a set of orthonormal vectors in $\mathbb{R}^d$ and F be a function in $C^{1,1}(\mathbb{R}^d)$. Then*

$$\|\widetilde{\nabla}^M_{\sigma,\boldsymbol{\Xi}}[F](\boldsymbol{x}) - \nabla F(\boldsymbol{x})\|^2 \tag{8}$$

$$\leq \frac{2C^2\pi d(M\,!)^2}{4^M((2M)\,!)^2}\sigma^{4M-2} + 32dL^2\sigma^2. \tag{9}$$

Proof. First, adapting [Nesterov and Spokoiny, 2017, Lemma 3] to 1-dimensional Gaussian smoothing, for any $\boldsymbol{\xi}$ being a unit vector in $\mathbb{R}^d$, there holds

$$|\mathscr{D}\,[G_\sigma(0\,|\,\boldsymbol{x},\boldsymbol{\xi})] - \nabla_{\boldsymbol{\xi}} F(\boldsymbol{x})| \leq 4\sigma L. \tag{10}$$

From (7) and (10), we have

$$\begin{aligned}
&\Big|\widetilde{\mathscr{D}}^M[G_\sigma(0\,|\,\boldsymbol{x},\boldsymbol{\xi}_i)] - \nabla_{\boldsymbol{\xi}_i} F(\boldsymbol{x})\Big|^2 \\
&\leq 2\Big|(\widetilde{\mathscr{D}}^M - \mathscr{D})[G_\sigma]\Big|^2 + 2\,|\mathscr{D}\,[G_\sigma] - \nabla_{\boldsymbol{\xi}_i} F(\boldsymbol{x})|^2 \\
&\leq \frac{2C^2(M\,!)^2\pi}{4^M((2M)\,!)^2}\,\sigma^{4M-2} + 32\sigma^2L^2, \;\; \forall i \in \{1,\ldots,d\}.
\end{aligned} \tag{11}$$

Summing (11) from $i = 1$ to d gives (9).

Let N be the total number of function evaluations. In our DGS algorithm, $N = Md$. An immediate consequence of Proposition 1 is that given a positive ε, $\sigma \leq \varepsilon/(4L\sqrt{d})$ and $N \geq d\log(2d/\varepsilon^2)$ are sufficient to obtain $\|\widetilde{\nabla}^M_{\sigma,\boldsymbol{\Xi}}[F] - \nabla F\| \leq \varepsilon$. Now, we compare these with the condition on N such that $\|g(\boldsymbol{x}) - \nabla F(\boldsymbol{x})\| \leq \varepsilon$, where $g(\boldsymbol{x})$ is an MC-based gradient estimator for ∇F_σ. For simplicity, we focus on the condition for $\|g(\boldsymbol{x}) - \nabla F_\sigma(\boldsymbol{x})\| < \varepsilon$, which is actually weaker because it does not count for the discrepancy between $\nabla F_\sigma(\boldsymbol{x})$ and $\nabla F(\boldsymbol{x})$. Let $\mathrm{Var}[g(\boldsymbol{x})]$ be the variance of g, applying Chebyshev inequality, one has

$$\mathbb{P}\left(\|g(\boldsymbol{x}) - \nabla F_\sigma(\boldsymbol{x})\| > \varepsilon\right) \leq \frac{d\,\mathrm{Var}[g(\boldsymbol{x})]}{\varepsilon^2},$$

therefore, $\|g(\boldsymbol{x}) - \nabla F_\sigma(\boldsymbol{x})\| \leq \varepsilon$ with probability exceeding $1-\eta$ given that $\mathrm{Var}[g(\boldsymbol{x})] < \eta\varepsilon^2/d$. It can be shown for forward finite-difference MC gradient estimator that $\mathrm{Var}[g(\boldsymbol{x})] \simeq \|\nabla F(\boldsymbol{x})\|^2/N$, see [Berahas et al., 2019], thus, this basic estimator requires $N = O(\frac{d\|\nabla F(\boldsymbol{x})\|^2}{\eta\varepsilon^2})$. It is possible to reduce the number of function evaluations with various variance reduction techniques, such as antithetic sampling, orthogonalization, control variates. In particular, if the variance of MC estimator is $\kappa\mathrm{Var}[g(\boldsymbol{x})]$ instead of $\mathrm{Var}[g(\boldsymbol{x})]$ for some $\kappa < 1$, then $N = O(\frac{\kappa d\|\nabla F(\boldsymbol{x})\|^2}{\eta\varepsilon^2})$ is sufficient. However, the proven rate of reduction κ is either independent or $O(1)$ on δ and ε for most variance reduction techniques, see, e.g., [Choromanski et al., 2018, Tang et al., 2019], in which cases, the theoretical dependence of N on d and ε cannot be relaxed.

With the gradient estimate in Proposition 1, we proceed to establish an error analysis for DGS in local regime. We show here a convergence rate of our method in optimizing strongly convex functions.

Proposition 2 *Let F be a strongly convex function in $C^{1,1}(\mathbb{R}^d)$, $\boldsymbol{x}^*$ be the global minimizer of F and the sequence $\{\boldsymbol{x}_t\}_{t\geq 0}$ be generated by Algorithm 1 with $\lambda = 1/(8L)$. Then, for any $t \geq 0$,*

$$F(\boldsymbol{x}_t) - F(\boldsymbol{x}^*) \tag{12}$$

$$\leq \frac{1}{2}L\left[\delta_\sigma + \left(1 - \frac{\tau}{16L}\right)^t\left(\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2 - \delta_\sigma\right)\right]. \tag{13}$$

where

$$\delta_\sigma = \left(\frac{128}{\tau^2} + \frac{16}{\tau L}\right)L^2 d\sigma^2 + \left(\frac{8}{\tau^2} + \frac{1}{2\tau L}\right)\frac{C^2(M\,!)^2\pi d}{4^M((2M)\,!)^2}\sigma^2. \tag{14}$$

Proof. First, we derive an upper bound for $\|\widetilde{\nabla}^M_{\sigma,\boldsymbol{\Xi}}[F](\boldsymbol{x})\|$. Recall

$$\|\widetilde{\nabla}^M_{\sigma,\boldsymbol{\Xi}}[F](\boldsymbol{x})\|^2 = \sum_{i=1}^d \Big|\widetilde{\mathscr{D}}^M\,[G_\sigma(0\,|\,\boldsymbol{x},\boldsymbol{\xi}_i)]\Big|^2. \tag{15}$$

Each term inside this sum can be bounded as

$$\begin{aligned}
&\Big|\widetilde{\mathscr{D}}^M\,[G_\sigma(0\,|\,\boldsymbol{x},\boldsymbol{\xi}_i)]\Big|^2 \leq 2\,|\mathscr{D}\,[G_\sigma(0\,|\,\boldsymbol{x},\boldsymbol{\xi}_i)]|^2 \\
&+2\Big|\widetilde{\mathscr{D}}^M\,[G_\sigma(0\,|\,\boldsymbol{x},\boldsymbol{\xi}_i)] - \mathscr{D}\,[G_\sigma(0\,|\,\boldsymbol{x},\boldsymbol{\xi}_i)]\Big|^2 \\
&\leq 64\sigma^2L^2 + 4|\nabla_{\boldsymbol{\xi}_i}F(\boldsymbol{x})|^2 + \frac{2C^2(M\,!)^2\pi}{4^M((2M)\,!)^2}\sigma^{4M-2}.
\end{aligned}$$

Plugging this into (15) gives

$$\|\widetilde{\nabla}^M_{\sigma,\boldsymbol{\Xi}}[F](\boldsymbol{x})\|^2 \leq 64dL^2\sigma^2 \tag{16}$$

$$+ 4\sum_{i=1}^{d} |\nabla_{\boldsymbol{\xi}_i} F(\boldsymbol{x})|^2 + \frac{2C^2\pi d(M\,!)^2}{4^M((2M)\,!)^2}\sigma^{4M-2}. \tag{17}$$

Denote $r_t = \|\boldsymbol{x}_t - \boldsymbol{x}^*\|$. Then

$$\begin{aligned} r_{t+1}^2 =& \|\boldsymbol{x}_t - \lambda\widetilde{\nabla}^M_{\sigma,\boldsymbol{\Xi}}[F](\boldsymbol{x}_t) - \boldsymbol{x}^*\|^2 \\ =& r_t^2 - 2\lambda\langle\widetilde{\nabla}^M_{\sigma,\boldsymbol{\Xi}}[F](\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{x}^*\rangle \\ &+ \lambda^2\|\widetilde{\nabla}^M_{\sigma,\boldsymbol{\Xi}}[F](\boldsymbol{x}_t)\|^2 \\ =& r_t^2 - 2\lambda\langle\widetilde{\nabla}^M_{\sigma,\boldsymbol{\Xi}}[F](\boldsymbol{x}_t) - \nabla F(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{x}^*\rangle \\ &- 2\lambda\langle\nabla F(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{x}^*\rangle \\ &+ 4\lambda^2\sum_{i=1}^{d}|\nabla_{\boldsymbol{\xi}_i}F(\boldsymbol{x}_t)|^2 + 64\lambda^2L^2d\sigma^2 \\ &+ \frac{2C^2\lambda^2(M\,!)^2\pi d}{4^M((2M)\,!)^2} + \sigma^{4M-2}. \end{aligned} \tag{18}$$

We proceed to bound the right hand side of (18). First, since F is strongly convex,

$$\begin{aligned} -2\lambda\langle\nabla F(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{x}^*\rangle \leq& 2\lambda F(\boldsymbol{x}^*) \\ &- 2\lambda F(\boldsymbol{x}_t) - \lambda\tau\|\boldsymbol{x}^* - \boldsymbol{x}_t\|^2. \end{aligned} \tag{19}$$

On the other hand,

$$\begin{aligned} &- 2\lambda\langle\widetilde{\nabla}^M_{\sigma,\boldsymbol{\Xi}}[F](\boldsymbol{x}_t) - \nabla F(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{x}^*\rangle \\ \leq& \frac{2\lambda}{\tau}\|\widetilde{\nabla}^M_{\sigma,\boldsymbol{\Xi}}[F](\boldsymbol{x}_t) - \nabla F(\boldsymbol{x}_t)\|^2 + \frac{\lambda\tau}{2}\|\boldsymbol{x}_t - \boldsymbol{x}^*\|^2 \\ \overset{(9)}{\leq}& \frac{4\lambda}{\tau}\cdot\frac{C^2\pi d(M\,!)^2}{4^M((2M)\,!)^2}\sigma^{4M-2} + \frac{64\lambda}{\tau}L^2d\sigma^2 + \frac{\lambda\tau}{2}\|\boldsymbol{x}_t - \boldsymbol{x}^*\|^2. \end{aligned} \tag{20}$$

Applying an estimate for convex, $C^{1,1}$-functions, see, e.g., [Nesterov, 2004], gives

$$\begin{aligned} 4\lambda^2\sum_{i=1}^{d}|\nabla_{\boldsymbol{\xi}_i}F(\boldsymbol{x}_t)|^2 =& 4\lambda^2\|\nabla F(\boldsymbol{x}_t)\|^2 \\ \leq& 8\lambda^2L(F(\boldsymbol{x}_t) - F(\boldsymbol{x}^*)). \end{aligned} \tag{21}$$

Combining (18)–(21), there holds

$$\begin{aligned} r_{t+1}^2 \leq& r_t^2 - (2\lambda - 8\lambda^2L)(F(\boldsymbol{x}_t) - F(\boldsymbol{x}^*)) \\ &+ \left(\frac{64\lambda}{\tau} + 64\lambda^2\right)L^2d\sigma^2 \\ &+ \left(\frac{4\lambda}{\tau} + 2\lambda^2\right)\frac{C^2(M\,!)^2\pi d}{4^M((2M)\,!)^2}\sigma^{4M-2}. \end{aligned} \tag{22}$$

Since F is a strongly convex function, for $\lambda = 1/(8L)$ we have that

$$-(2\lambda - 8\lambda^2L)(F(\boldsymbol{x}_t) - F(\boldsymbol{x}^*)) \leq \frac{\tau}{16L}\|\boldsymbol{x}_t - \boldsymbol{x}^*\|^2. \tag{23}$$

Assuming $\sigma < 1$. We derive from (22), (23) and (14) that $r_{t+1}^2 - \delta_\sigma \leq \left(1 - \frac{\tau}{16L}\right)(r_t^2 - \delta_\sigma)$, which yields

$$r_t^2 - \delta_\sigma \leq \left(1 - \frac{\tau}{16L}\right)^t (r_0^2 - \delta_\sigma).$$

Note that $F(\boldsymbol{x}_t) - F(\boldsymbol{x}^*) \leq \frac{1}{2}L\|\boldsymbol{x}_t - \boldsymbol{x}^*\|^2$, since $f \in C^{1,1}(\mathbb{R}^d)$, we arrive at the conclusion. $\square$

It is worth remarking a few things on the above proposition. First, we obtain the global linear rate of convergence with DGS, which is expected for strongly convex functions. Second, the result allows random perturbation of $\boldsymbol{\Xi}$ as long as $\boldsymbol{\Xi}$ remains orthonormal. Finally, this proposition proves the scalability of our algorithm in the strongly convex setting. In particular, to guarantee $F(\boldsymbol{x}_t) - F(\boldsymbol{x}^*) \leq \varepsilon$, from error estimate (13), we need to choose

$$\sigma \leq O\left(\sqrt{\frac{\varepsilon}{d}}\right), \text{\# fun evals} = dM \geq O\left(d\log\left(\frac{d}{\varepsilon}\right)\right),$$

$$\text{\# iterations } = O\left(\log\frac{1}{\varepsilon}\right).$$

This indicates that the number of iterations required by our approach is completely independent of the dimension, while the total number of function evaluations is only slightly higher than nonparallelizable random search approach, e.g., [Nesterov and Spokoiny, 2017, Bergou et al., 2019].

The above discussion shows that the performance of the DGS method is consistent with the local gradient estimation methods when σ is small, which paves the way for further analysis in the nonlocal regime for which the DGS gradient is designed for.

References

Y. Akimoto and N. Hansen. Projection-based restricted covariance matrix adaptation for high dimension. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, page 197–204. Association for Computing Machinery, 2016. ISBN 9781450342063.

Erik Andreassen, Anders Clausen, Mattias Schevenels, Boyan S Lazarov, and Ole Sigmund. Efficient topology optimization in matlab using 88 lines of code. *Structural and Multidisciplinary Optimization*, 43(1):1–16, 2011.

A. Auger and N. Hansen. A restart cma evolution strategy with increasing population size. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1769–1776 Vol. 2, 2005.

M. Bendsøe. Optimal shape design as a material distribution problem. *Structural optimization*, 1(4):193–202, 1989.

Martin Philip Bendsoe and Ole Sigmund. *Topology optimization: theory, methods, and applications*. Springer Science & Business Media, 2013.

A. S. Berahas, L. Cao, K. Choromanskiv, and K. Scheinberg. A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *arXiv:1905.01332*, 2019.

E. Bergou, E. Gorbunov, P. Richtárik, and P. Richtárik. Stochastic three points method for unconstrained smooth minimization. *arXiv: 1902.03591*, 2019.

K. Choromanski, M., V. Sindhwani, R. Turner, and A. Weller. Structured evolution with compact architectures for scalable policy optimization. *International Conference on Machine Learning*, pages 969–977, 2018.

K. Choromanski, A. Pacchiano, J. Parker-Holder, Y. Tang, and V. Sindhwani. From complexity to simplicity: Adaptive es-active subspaces for blackbox optimization. In *Advances in Neural Information Processing Systems 32*, pages 10299–10309. Curran Associates, Inc., 2019.

D. Eriksson, M. Pearce, J. Gardner, R. Turner, and M. Poloczek. Scalable global optimization via local bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 5496–5507, 2019.

David Eriksson, Kun Dong, Eric Hans Lee, David Bindel, and Andrew Gordon Wilson. Scaling gaussian process regression with derivatives. In *NeurIPS*, 2018.

J. Gardner, M. Kusner, Z. Xu, K. Weinberger, and J. Cunningham. Bayesian optimization with inequality constraints. In *ICML*, pages 937–945, 2014.

James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

I. Loshchilov, T. Glasmachers, and H. Beyer. Large scale black-box optimization by limited-memory matrix adaptation. *IEEE Transactions on Evolutionary Computation*, 23(2):353–358, 2019.

L. Miranda. Pyswarms: a research toolkit for particle swarm optimization in python. *Journal of Open Source Software*, 3(21):433, 2018.

Yurii Nesterov. *Introductory Lectures on Convex Optimization*. Springer US, 2004.

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

M. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2):155–162, 1964.

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

Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Ole Sigmund. Morphology-based black and white filters for topology optimization. *Structural and Multidisciplinary Optimization*, 33(4-5):401–424, 2007.

R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. of global optimization*, 11(4):341–359, 1997.

K. Svanberg. The method of moving asymptotes—a new method for structural optimization. *International J. for numerical methods in engineering*, 24(2):359–373, 1987.

Yunhao Tang, Krzysztof Choromanski, and Alp Kucukelbir. Variance reduction for evolution strategies via structured control variates. *ArXiv*, abs/1906.08868, 2019.

Jian Wu, Matthias Poloczek, Andrew Gordon Wilson, and Peter I Frazier. Bayesian optimization with gradients. *Advances in Neural Information Processing Systems*, 2017: 5268–5279, 2017.