BOING: Optimizing Bayesian Optimization with Information Geometry

Anonymous authors
Paper under double-blind review

Abstract

We study Bayesian optimization (BO) through the lens of information geometry. We use insights from existing work that the surrogate, conventionally a GP, maps the input space to a lower-dimensional statistical manifold. The Acquisition Functions (AFs) in BO are defined on this manifold. Instead of conventional gradient-based optimization methods like L-BFGS or Adam, we propose BOING: an iteration-free, trust-region-based method for AF optimization on a low-dimensional manifold that is parallelizable on a GPU. For a certain class of AFs in sequential BO, we show that the optimization of AF can be performed on a 1-D search space. We use the Fisher Information Matrix to construct the trust regions for BOING. Our method achieves competitive or superior overall BO performance. Our method provides a 30 - 20× computational speedup for low-dimensional BO and reduces the total BO time by 20 - 40%. For high-dimensional BO, we achieve a 50 - 100× speedup and reduce the total BO time by 300 - 500%. We show speedups across multiple benchmarks and acquisition functions.

1 Introduction

Bayesian Optimization (BO) is a popular framework for optimizing black-box functions with expensive evaluations (Frazier, 2018; Mockus, 1998). It has proven to be effective in hyperparameter tuning (Wu et al., 2019), robotics policy search (Jaquier et al., 2020), and engineering design (Lam et al., 2018). As explained in Section 3, at each iteration t, BO alternates between fitting a surrogate $\mathcal{M}_t$ to the observed data $\mathcal{D}_t$ and optimizing an Acquisition Function (AF) $\alpha(\cdot)$ to select the next input candidate $\mathbf{x}_{t+1}$.

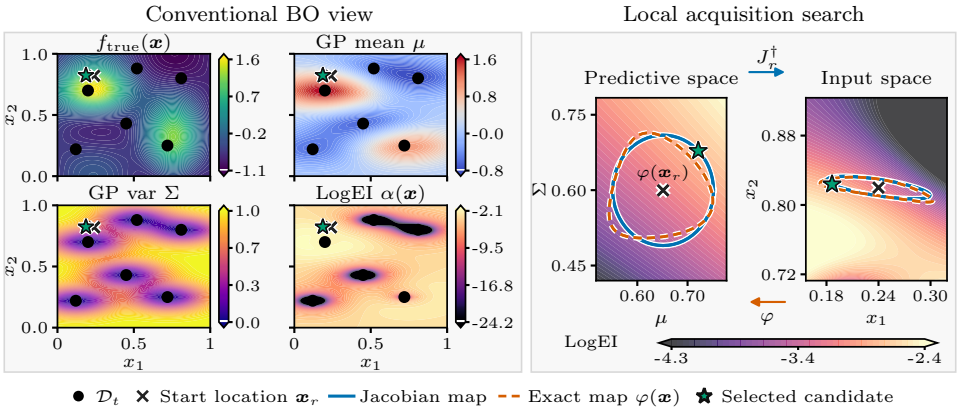


Figure 1: Left: objective, Gaussian process (GP) moments and LogEI. Right: LogEI around the input location $\mathbf{x}_r$ marked with cross. Blue shows the Fisher ellipse at predictive parameter $\varphi(\mathbf{x}_r)$ and its Jacobian map $J_r^\dagger$ back into input space. Green star marks the best sampled predictive state and its mapped input. Orange shows the true posterior map φ.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

However, the inner optimization of AF can be challenging. This issue is well documented in the high-dimensional BO (HDBO) literature and proposed solutions include dimensionally scaled lengthscale priors (Hvarfner et al., 2024; Xu et al., 2024), Random Axis Aligned Subspace Perturbations (RAASP) (Papenmeier et al., 2025) and conformal mappings (Doumont et al., 2025).

We view HDBO through the lens of information geometry in Section 3 and 4. As shown in Figure 1, the surrogate $\mathcal{M}_t$ is a map from input space $\mathcal{X}$ to the statistical manifold $\mathcal{S}$ of predictive distributions. These predictive distribution parameters in $\mathcal{S}$, like mean and variance, are the parameters that the AF really depends on. Information geometry (Amari, 2016; Lee, 2018) provides a natural framework for studying how the surrogate induces geometry on the input space through the pullback Fisher tensor. This is view is also discussed by De Ath et al. (2021); Kiroriwal et al. (2026).

In this work, our main contributions are as follows:

- In Section 4, we propose BOING, a novel iteration-free, Fisher Information-constrained local optimization method on the induced manifold for AF optimization in BO. BOING uses information geometry to optimize the AF in the predictive space.
- In Section 4.2, we show that for a certain class of AFs in sequential BO, the AF optimization, originally in D_x, can be reduced to a search over a 1-D simplex.
- As shown in Section 5, BOING results in *multifold* acquisition-optimization speedups over the tested iterative methods while maintaining competitive or superior BO performance.

2 Related Work

High-dimensional BO and local search. Hvarfner et al. (2024); Xu et al. (2024); Papenmeier et al. (2025) independently reported that dimensionally scaled priors for lengthscale initialization can improve high-dimensional GP modeling and acquisition search. RAASP and related coordinate perturbations provide starting locations around observed inputs (Regis & Shoemaker, 2013; Rashidi et al., 2024; Papenmeier et al., 2025) to help avoid vanishing gradients. Eriksson et al. (2019) introduced TuRBO, a trust-region-based BO, in which they optimize using gradient-based methods over a smaller subset of the full search space $\mathcal{X}$.

Using geometry of the input domain. Jaquier et al. (2020); Jaquier & Rozo (2020) study BO on manifolds and lower-dimensional nested manifolds. Jaquier et al. (2022) construct Riemannian Matérn kernels for domains such as spheres, rotation groups, and positive-definite matrices. Pavesi et al. (2026) use information geometry to construct Matérn kernels and acquisition optimizers over the probability simplex.

Acquisition search through predictive moments. De Ath et al. (2021) analyzed the dependence of AF on the predictive parameters and showed that EI and UCB select from the mean–uncertainty Pareto front, whereas PI does not. Carciaghi et al. (2025) use posterior mean and variance gradients for Batch-BO. Yu et al. (2025) also proposes to use the mean gradient for creating a subspace for Thompson sampling. Whitehead (2026) shows that EI and LogEI optimization for the spherical linear GP of Doumont et al. (2025) reduces to a one-dimensional search over the mean–variance Pareto front on the unconstrained sphere.

Posterior maps and pullback geometry. Pulling a metric through a probabilistic model is well established, and Tosi et al. (2014) study expected metrics for Gaussian-process latent variable models. Arvanitidis et al. (2022) pull the Fisher–Rao metric of decoder distributions back to a latent space. Rozo et al. (2025) study the pullback metric induced by wrapped Gaussian-process latent variable models for manifold-valued data. Amari (2016) develops the Fisher geometry of statistical models, and Martens (2020) relates natural-gradient updates to local Kullback–Leibler (KL)-constrained optimization.

Acquisition-gradient bounds and local trust regions. Kiroriwal et al. (2026) derive an AF-gradient bound, Proposition 4.2, that separates acquisition sensitivity from surrogate-induced geometry and provides a common interpretation of dimension-scaled lengthscales and RAASP in HDBO.

They introduce FITR, which replaces lengthscale-based trust-region weights in TuRBO with Fisher Information Matrix (FIM)-based sensitivities that are location-dependent.

Fisher SAM. Fisher SAM (Kim et al., 2022) uses a Fisher ellipsoid and a first-order approximation of the objective function for parameter perturbations. The authors report improved generalization and robustness in neural network training at a computational cost comparable to SAM. The radius remains a hyperparameter, with mild sensitivity except at overly large values in their experiments.

3 PRELIMINARIES

Surrogate Models We maximize f_{true} on $\mathcal{X} \subseteq \mathbb{R}^{D_x}$ using the observed data $\mathcal{D}_t = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$. At each iteration t, the surrogate model $\mathcal{M}_t$ induces a *posterior map* $\varphi_t : \mathcal{X} \to \mathcal{S}$ where $\varphi_t(\mathbf{x}) = \boldsymbol{\theta}_{t,\mathbf{x}}$. These are the parameters of the latent predictive distribution, whose density is $p(y; \boldsymbol{\theta}_{t,\mathbf{x}})$. The hyperparameters $\boldsymbol{\lambda}_t$ of $\mathcal{M}_t$ can be trained by maximizing the marginal log-likelihood Rasmussen (2003) or another suitable objective Blei et al. (2017); Moss et al. (2023). As shown in the manifold view in Figure 1, the surrogate model can also be interpreted as a smooth map from inputs to a family of probability distributions. Each input $\mathbf{x} \in \mathcal{X}$ is mapped to a point $\boldsymbol{\theta}_{t,\mathbf{x}}$ on the statistical manifold $\mathcal{S}$, the chosen family of predictive distributions Amari (2016). This map is at least C^1 differentiable, a mild condition satisfied by Gaussian process (GP) models with a continuously differentiable kernel. For a scalar output $y \in \mathbb{R}$ at an input location $\mathbf{x} \in \mathcal{X}$, $\mathcal{M}_t$ models $\boldsymbol{\theta}_{t,\mathbf{x}} = (\mu_{t,\mathbf{x}}, \Sigma_{t,\mathbf{x}})$ and gives $p(f(\mathbf{x})|\mathcal{D}_t) = \mathcal{N}(\mu_{t,\mathbf{x}}, \Sigma_{t,\mathbf{x}})$ with $\mathcal{S} = \mathbb{R} \times \mathbb{R}_+$ Rasmussen (2003). As discussed in Appendix A, μ is the latent predictive mean, Σ is its variance, and $\sqrt{\Sigma}$ is its standard deviation throughout the paper.

Acquisition Function (AF) An AF α is expected utility at a location $\mathbf{x} \in \mathcal{X}$ and is defined as $\alpha(\mathbf{x}; \boldsymbol{\xi}) = \mathbb{E}_y[a(y; \boldsymbol{\theta}_{t,\mathbf{x}}, \boldsymbol{\xi})]$, where a is the utility function, $\boldsymbol{\xi}$ are respective utility parameters Frazier (2018). For AFs that depend on the latent predictive distribution parameterized by $\boldsymbol{\theta}_{t,\mathbf{x}}$, we assume that α depends on $\mathbf{x}$ only through $\boldsymbol{\theta}_{t,\mathbf{x}}$ as $\alpha(\mathbf{x}; \boldsymbol{\xi}) = \bar{\alpha}(\varphi_t(\mathbf{x}); \boldsymbol{\xi})$, with $\boldsymbol{\xi}$ fixed during inner optimization.Wilson et al. (2017; 2018) show the reparameterized form of AF where $\bar{\alpha}(\boldsymbol{\theta}; \boldsymbol{\xi}) := \mathbb{E}_\zeta[a_\zeta(\boldsymbol{\theta}, \zeta; \boldsymbol{\xi})]$, where $\zeta \sim \mathcal{N}(0, 1)$ and a_ζ is reparmeterized utility. Finite Monte Carlo estimates use fixed base samples during inner optimization; acquisitions requiring joint predictions use the corresponding joint predictive state. Appendix B summarizes common AFs and Algorithm 2 discusses the generic BO loop where L-BFGS or Adam are conventional AF optimization method and $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x} \in \mathcal{X}} \bar{\alpha}(\varphi_t(\mathbf{x}))$.

Fisher Information Matrix (FIM) and Pullback Tensor For a parametric distribution family $p(y; \boldsymbol{\theta})$ with $\boldsymbol{\theta} \in \mathcal{S}$, the FIM $\mathcal{I}(\boldsymbol{\theta}) \in \mathbb{R}^{D_\theta \times D_\theta}$ is the expected outer product of $\nabla_{\boldsymbol{\theta}} \log p(y; \boldsymbol{\theta})$ and quantifies the sensitivity of the predictive distribution $p(y; \boldsymbol{\theta})$ to changes in $\boldsymbol{\theta}$ (Amari & Nagaoka, 2000; Amari, 2016). It is defined as

$$[\mathcal{I}(\boldsymbol{\theta})]_{ij} = \mathbb{E}_{p(y;\boldsymbol{\theta})}\left[\frac{\partial \log p(y; \boldsymbol{\theta})}{\partial \theta_i} \frac{\partial \log p(y; \boldsymbol{\theta})}{\partial \theta_j}\right]. \tag{1}$$

4 BOING: FISHER-CONSTRAINED ACQUISITION OPTIMIZATION

We propose BOING, an iteration-free AF optimization method that uses R constrained local optimizations on the predictive manifold $\mathcal{S}$. For a given radius, each local problem maximizes the AF in the predictive manifold $\mathcal{S}$ around the predictive parameter $\boldsymbol{\theta}_r$ for a given starting input $\mathbf{x}_r$, subject to a Fisher constraint. Algorithm 1 shows BOING for univariate GP as a surrogate for sequential BO in detail. BOING serves as an AF optimization method that can be used in BO (Algorithm 2).

We build on the findings of Fisher SAM (Kim et al., 2022) and the acquisition-gradient bound of Kiroriwal et al. (2026) (Proposition 4.2), which motivate the use of Fisher constraints. FITR shapes the input-space trust region, while BOING addresses the inner AF optimization. Since we discuss the AF optimization at a BO iteration t, and for ease of notation, we drop the index t because it is fixed during the inner optimization.

Assumption 4.1 (Surrogate Regularity)**.** *The posterior map φ is at least C^1 on $\mathcal{X}$ and $\mathcal{I}(\varphi(\mathbf{x})) \succ 0$ for all $\mathbf{x} \in \mathcal{X}$ (a positive kernel diagonal and observation noise $\sigma_\varepsilon^2 > 0$ ensure positive latent variance for GP regression).*

Definition 4.1 (Pullback Fisher Tensor)**.** *Under Assumption 4.1, the pullback Fisher tensor induced on the input space $\mathcal{X}$ by the predictive distribution manifold $\mathcal{S}$ through the posterior map φ is*

$$\mathbf{g}(\mathbf{x}) = J(\mathbf{x})^\top \mathcal{I}\big(\varphi(\mathbf{x})\big)\, J(\mathbf{x}) \;\in\; \mathbb{R}^{D_x \times D_x}, \tag{2}$$

where $\mathbf{J}(\mathbf{x}) = \partial \boldsymbol{\theta}_\mathbf{x} / \partial \mathbf{x} \in \mathbb{R}^{D_\theta \times D_x}$ is the Jacobian of φ at $\mathbf{x}$.

Proposition 4.2 (Acquisition Gradient Bound)**.** *Let φ and $\bar{\alpha}$ be continuously differentiable, with $\mathcal{I}(\varphi(\mathbf{x})) \succ 0$. For any $\mathbf{x} \in \mathcal{X}$, let $\boldsymbol{\theta}_\mathbf{x} = \varphi(\mathbf{x})$. Then*

$$\big\|\nabla_\mathbf{x}\, \alpha(\mathbf{x})\big\|_2 \;\le\; \sqrt{C_\alpha(\boldsymbol{\theta}_\mathbf{x}) \cdot \mathrm{tr}(\mathbf{g}(\mathbf{x}))}, \tag{3}$$

where the acquisition sensitivity *is*

$$C_\alpha(\boldsymbol{\theta}_\mathbf{x}) = \nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_\mathbf{x})^\top\, \mathcal{I}(\boldsymbol{\theta}_\mathbf{x})^{-1} \nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_\mathbf{x})\,. \tag{4}$$

4.1 Local Optimization in Predictive Space

The Fisher constraint limits the change in the predictive distribution. For each starting location $\mathbf{x}_r$, let $\boldsymbol{\theta}_r = \varphi(\mathbf{x}_r)$, $\mathcal{I}_r := \mathcal{I}(\boldsymbol{\theta}_r)$, $J_r = J(\mathbf{x}_r)$ and $\mathbf{g}_r := J_r^\top \mathcal{I}_r J_r$. The quadratic constraint in (5) uses the local curvature of KL divergence (Amari, 2016; Martens, 2020), which can be defined using a radius κ and FIM. In Section 4.3 we discuss how to select a radius.

Multiple Local Fisher-constrained AF optimization for a given BO iteration t

1. Optimize on the predictive manifold. For a scalar $\kappa > 0$, each local constrained optimization $r \in \{1, \dots, R\}$ is defined as

$$\begin{aligned} \boldsymbol{\Delta\theta}_r^\star(\kappa) = \underset{(\boldsymbol{\theta}_r + \boldsymbol{\Delta\theta}) \in \mathcal{S}}{\arg\max}\; & \bar{\alpha}(\boldsymbol{\theta}_r + \boldsymbol{\Delta\theta}), \\ \text{subject to} \quad & \boldsymbol{\Delta\theta}^\top \mathcal{I}_r \boldsymbol{\Delta\theta} \le \kappa^2, \qquad \boldsymbol{\Delta\theta} \in \mathrm{range}(J_r). \end{aligned} \tag{5}$$

The update is defined as $\boldsymbol{\theta} := \boldsymbol{\theta}_r + \boldsymbol{\Delta\theta}$, and at optima $\boldsymbol{\theta}_r^\star(\kappa) := \boldsymbol{\theta}_r + \boldsymbol{\Delta\theta}_r^\star(\kappa)$.

2. Map back to input space. The updated optimal predictive parameter $\boldsymbol{\theta}_r^\star(\kappa)$ gives a first-order input displacement as

$$\boldsymbol{d}_r^\star(\kappa) := J_r^\dagger \boldsymbol{\Delta\theta}_r^\star(\kappa). \tag{6}$$

3. Select the next evaluation. The next input location $\mathbf{x}_{t+1}$ for BO is chosen using the candidate set $\mathcal{X}^{\mathrm{cand}}$ as

$$\mathbf{x}_{t+1} = \underset{\mathbf{x} \in \mathcal{X}^{\mathrm{cand}}}{\arg\max}\, \alpha(\mathbf{x}), \quad \text{where} \quad \mathcal{X}^{\mathrm{cand}} := \big\{\mathbf{x}_r + \boldsymbol{d}_r^\star(\kappa) \;: r = 1, \dots, R\big\}. \tag{7}$$

Since the rank of $\mathbf{g}_r$ is at most $\min(D_\theta, D_x)$, the null space is non-trivial when $D_x > D_\theta$. The second constraint ensures that the predictive change is in $\mathrm{range}(J_r)$ and can therefore be reached to first order from the input space. The pseudoinverse $J_r^\dagger$ ensures that the input displacement $\mathbf{d}$ is orthogonal to $\mathrm{null}(J_r)$, hence avoiding perturbations that do not change the predictive parameters to first order. Figure 1 compares the input curve obtained by applying $J_r^\dagger$ in (6) to displacements on the Fisher ellipse with the input contour obtained from the posterior map φ under the Fisher constraint.

Conventionally, AF optimization using iterative methods like L-BFGS (Liu & Nocedal, 1989) or Adam (Kingma & Ba, 2014) requires multiple posterior and gradient evaluations of the learned model $\mathcal{M}$. For a fixed GP at iteration t, the training-covariance factorization is cached and reused for posterior queries (Rasmussen, 2003). BoTorch (Balandat et al., 2020) can batch independent restarts on a GPU. However, successive optimizer updates depend on earlier values and gradients, so these evaluations remain sequential over the optimization iterations. Since BOING candidates are not generated iteratively, all posterior evaluations can be performed in a batch on a GPU, and repeated gradient evaluations are not required.

4.2 Solving the local optimization on the predictive manifold

Equations 6 and (7) are trivial to solve. However, (5) is not. We propose Assumptions 4.2 and 4.3 that help us solve the constrained local optimization on the manifold.

Assumption 4.2 (Local Coordinatewise Monotonicity)**.** *At each starting predictive parameter $\boldsymbol{\theta}_r$, there exists a local neighborhood*

$$\boldsymbol{\Delta\theta}^\top \mathcal{I}_r \boldsymbol{\Delta\theta} < (\kappa_r^{\text{loc}})^2, \quad \textit{such that} \quad \kappa_r^{\text{loc}} > 0 \quad \textit{and} \quad \boldsymbol{\theta}_r + \boldsymbol{\Delta\theta} \in \mathcal{S}$$

where $\bar{\alpha}$ is at least $\mathcal{C}^1$ with $(\partial_{\theta_i} \bar{\alpha})(\boldsymbol{\theta}_r + \boldsymbol{\Delta\theta}) > 0 \; \forall \; i \in \{1, \ldots, D_\theta\}$.

Assumption 4.3 (Full Row Rank of Jacobian)**.** *At each starting location $\mathbf{x}_r$, the* $\operatorname{rank}(J_r) = D_\theta$.

Proposition 4.3 (Maximum Lies on the Fisher Boundary)**.** *For $0 < \kappa < \kappa_r^{\text{loc}}$ and under Assumptions 4.1, 4.2, and 4.3, the selected displacement in* (7) *satisfies*

$$\left(\boldsymbol{d}_r^\star(\kappa)\right)^\top \mathbf{g}_r \boldsymbol{d}_r^\star(\kappa) = \kappa^2. \tag{8}$$

Proposition 4.3, discussed in detail in Appendix D.2, converts the inequality constraint in (5) into an equality constraint at the solution. This is important because the Fisher constraint is active under the stated local assumptions, and we need to search only for optima on the boundary of the constraint instead of the interior.

An implicit optimality condition. Since the Fisher constraint is active by Proposition 4.3, the Karush-Kuhn-Tucker (KKT) conditions give the necessary condition for the solution of (5):

$$\nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r^\star(\kappa)) = 2\eta_r^\star(\kappa) \mathcal{I}_r \boldsymbol{\Delta\theta}_r^\star(\kappa), \qquad \eta_r^\star(\kappa) > 0, \quad \text{hence}$$

$$\boldsymbol{\Delta\theta}_r^\star(\kappa) = \frac{\kappa \, \mathcal{I}_r^{-1} \nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r^\star(\kappa))}{\sqrt{\left[\nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r^\star(\kappa))\right]^\top \mathcal{I}_r^{-1} \nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r^\star(\kappa))}}. \tag{9}$$

The gradient $\nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r^\star(\kappa))$ is evaluated at the unknown optimal predictive parameter $\boldsymbol{\theta}_r^\star(\kappa)$, so (9) remains implicit. Appendix D.3 gives the complete KKT conditions and derivation. For comparison, Fisher SAM (Kim et al., 2022) linearises the nonlinear objective function at the starting location $\boldsymbol{\theta}_r$ using a first-order approximation (Kim et al., 2022, Eqs. 8–10) as shown in Appendix D.4. BOING evaluates the nonlinear AF on the predictive grid in (11), while reusing the starting Jacobian for the map back to input space in (6).

4.3 Simplex Search for AF Optimization

For a given constraint radius κ, we solve (5) by searching the Fisher boundary using the optimality condition in (9). For a scalar GP used in sequential BO, a predictive parameter is $\boldsymbol{\theta} = (\mu, \Sigma)^\top \in \mathbb{R} \times \mathbb{R}_+$ and the Fisher matrix is $\mathcal{I}(\boldsymbol{\theta}) = \operatorname{diag}(\Sigma^{-1}, (2\Sigma^2)^{-1})$. By Proposition 4.3 and Assumption 4.2, the constrained maximizer lies on the quarter of the Fisher ellipse with non-negative mean and variance direction $\boldsymbol{\theta}_r^\star(\kappa) \in \left\{ \boldsymbol{\theta}_r + \boldsymbol{\Delta\theta} \;\middle|\; \boldsymbol{\Delta\theta} \in \mathbb{R}^2_{\geq 0}, \quad \boldsymbol{\Delta\theta}^\top \mathcal{I}_r \boldsymbol{\Delta\theta} = \kappa^2 \right\}$.

Since both components of the acquisition gradient are positive throughout the local neighborhood, the normalized gradient for every predictive state in the neighborhood lies on the one-dimensional simplex (Boyd & Vandenberghe, 2004) defined as

$$\frac{\nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta})}{\|\nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta})\|_1} \in \left\{ \boldsymbol{\Omega} = \begin{bmatrix} \omega \\ 1 - \omega \end{bmatrix} : \omega \in [0, 1] \right\}. \tag{10}$$

The KKT condition (9) makes the gradient at a constrained maximum parallel to the outward ellipse normal. As discussed in Appendix D.5, using a simple search over the positive quadrant of the constraint ellipse with ω, the maximizer for each of the r local maximizations in (5) can be found using

1-D Simplex formulation of local Fisher-constrained problem in Equation 5

$$\boldsymbol{\Delta\theta}_r^\star(\kappa) := \boldsymbol{\Delta\theta}_r\left(\kappa, \omega_r^\star(\kappa)\right) \quad \text{where} \quad \omega_r^\star(\kappa) := \operatorname*{arg\,max}_{\omega \in [0,1]} \bar{\alpha}(\boldsymbol{\theta}_r(\kappa, \omega)); \tag{11}$$

$$\boldsymbol{\theta}_r(\kappa, \omega) := \boldsymbol{\theta}_r + \boldsymbol{\Delta\theta}_r(\kappa, \omega), \quad \text{where} \quad \boldsymbol{\Delta\theta}_r(\kappa, \omega) = \frac{\kappa \, \mathcal{I}_r^{-1} \boldsymbol{\Omega}}{\sqrt{\boldsymbol{\Omega}^\top \mathcal{I}_r^{-1} \boldsymbol{\Omega}}}. \tag{12}$$

For a univariate GP, substituting the diagonal Fisher matrix into (12) gives the mean and variance updates in (17) (Appendix C.2). Algorithm 1 uses this expanded form, so the Fisher scaling is retained without explicitly constructing or inverting $\mathcal{I}_r$.

Since the search space in (11) is 1-D, to keep the algorithm simple and parallelizable, we approximate it by simple grid search with N_ω samples over $[0, 1]$, as shown in Figure 2.

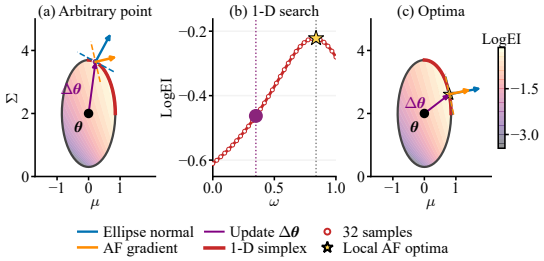


Figure 2: Approximate $N_\omega = 32$-point search for LogEI $\bar{\alpha}$ at $(\mu_r, \Sigma_r) = (0, 2)$ and $\kappa = 0.6$. At the true optimum, the constraint normal (blue) is parallel to the AF gradient; the grid approximates this alignment.

Starting locations and radii. The choice of starting locations $\mathbf{x}_r$ and radius κ determines the local neighborhoods searched by BOING. Using the insights from Proposition 4.2, FITR (Kiroriwal et al., 2026), and RAASP (Papenmeier et al., 2025), we use a 1:1 ratio of local perturbations around the best-observed locations and uniformly sampled points from the input space $\mathcal{X}$.

For AFs such as EI and UCB, Assumption 4.2 follows directly. However, for AFs such as GIBBON (Moss et al., 2021), we cannot prove the assumption and evaluate this locally motivated search empirically in Section 5. Even when monotonicity holds, (6) maps the predictive update defined by (12) back to input space only to first order. Figure 1 shows the difference between this Jacobian approximation and the true posterior.

Having solved the (9) using Equations 11 and 12, the choice of radius still remains an open question. We propose a radius $\kappa := c\overline{\lambda}$, that is dependent on the mean lengthscale $\overline{\lambda} := \text{mean}(\boldsymbol{\lambda})$ of the $\mathcal{M}_t$, with a radius multiplier $c \in \mathcal{C}_{\text{grid}} \subseteq [0, 1]$. A small radius restricts exploration, while a large radius can produce wasted candidates because the approximation becomes less accurate. Since $\kappa = c\overline{\lambda}$, we can also pose the search for optimal radius as a search over $\mathcal{C}_{\text{grid}}$ for $c^\star$; as shown in Figure 1.

Searching over the radius. Equations (11)–(12) give one predictive update for each fixed radius. Figure 3 shows AFs around the starting location $\boldsymbol{x}$ using both the Jacobian map and the true posterior map. Since the Jacobian map is affine, it might appear that the maximum radius multiplier c is the optimal choice. However, the exact AF value can differ from the Jacobian approximation. The "AF change" plot shows how the true AF value differs from the Jacobian approximation as the radius grows. The best AF value occurs at an intermediate radius, although the predicted optimum keeps increasing.

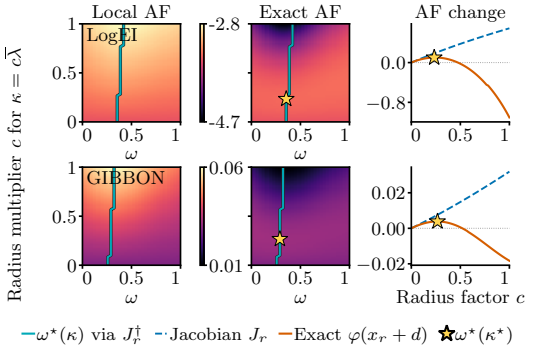


Figure 3: LogEI (top) and GIBBON (bottom). Left/middle: AF values under the Jacobian and exact posterior maps. Teal shows the optimal ω for each respective radius multiplier c. Right: AF change from the starting location.

Since the optimization of (11) and (12) is on a 1-D simplex and the FIM is 2-dimensional, each local optimization is computationally cheap. We propose evaluating N_c radii multipliers with each c in $[0, \ldots, 1]$. Figure 3 shows the search over the $\mathcal{W}_{\text{grid}} \times \mathcal{C}_{\text{grid}}$ for two AFs, along with the resulting changes in AF relative to the starting location.

Candidate selection and parallel evaluation. For each starting location $\mathbf{x}_r$, BOING first computes its predictive state and posterior Jacobian J_r. For each radius multiplier $c \in \mathcal{C}_{\text{grid}}$, it evaluates the finite simplex grid $\mathcal{W}_{\text{grid}}$ and retains the coefficient $\omega_r^\star$ that gives the largest approximate AF using (11). The corresponding predictive update is mapped back to input space using (6), producing one candidate for each starting-location–radius pair. All searches over ω, radii, and starting locations are conditionally independent given the fitted GP and can therefore be evaluated in parallel as GPU batches. Finally, BOING evaluates the true AF values at the mapped candidates $\mathcal{X}^{\text{cand}}$ using the true GP posterior. Maximizer $\mathbf{x}_{t+1}$ is then selected from the mapped candidates $\mathcal{X}^{\text{cand}}$ according to (7). Algorithm 1 summarizes this complete candidate-generation and selection procedure.

Algorithm 1: BOING for a scalar GP and for sequential AF at BO iteration t

Input: GP $\mathcal{M}_t$; AF $\bar{\alpha}$; Input space $\mathcal{X}$; N_ω; Starting locations $\{\mathbf{x}_r\}_{r=1}^{R} \subseteq \mathcal{X}$ (Appendix C.1); Radius multipliers $\mathcal{C}_{\text{grid}} \subset [0,1]$ with $|\mathcal{C}_{\text{grid}}| = N_c$ (Appendix C.1)

Output: Next input location $\mathbf{x}_{t+1}$

$\mathcal{X}^{\text{cand}} \leftarrow \{\mathbf{x}_r\}_{r=1}^{R}$

$\mathcal{W}_{\text{grid}} \leftarrow \text{linspace}(0, 1, N_\omega), \quad N_\omega \geq 2$

foreach $r \in \{1, \ldots, R\}$ **do** ▷ GPU parallelizable

  $\boldsymbol{\theta}_r \leftarrow \varphi_t(\mathbf{x}_r)$

  $J_r \leftarrow \frac{\partial \boldsymbol{\theta}_r}{\partial \mathbf{x}_r^\top}$ ▷ two batched backward passes

foreach $(r, c) \in \{1, \ldots, R\} \times \mathcal{C}_{\text{grid}}$ **do** ▷ parallel on GPU

  $\kappa_t(c) \leftarrow c\overline{\lambda}_t$ ▷ Equation (15)

  $\omega^\star_{r,c} \leftarrow \arg\max_{\omega \in \mathcal{W}_{\text{grid}}} \bar{\alpha}(\boldsymbol{\theta}_r(\kappa_t(c), \omega))$ ▷ Equations (18)–(19)

  $\boldsymbol{d}^\star_{r,c} \leftarrow J_r^\dagger \boldsymbol{\Delta\theta}_r(\kappa_t(c), \omega^\star_{r,c})$

  $\mathbf{x}^\star_{r,c} \leftarrow \mathbf{x}_r + \boldsymbol{d}^\star_{r,c}$ ▷ Equation (21)

$\mathcal{X}^{\text{cand}} \leftarrow \mathcal{X}^{\text{cand}} \cup \Big(\bigcup_{r=1}^{R} \{\mathbf{x}^\star_{r,c} \; \forall \; c \in \mathcal{C}_{\text{grid}}\} \Big)$ ▷ Equation (22)

return $\arg\max_{\mathbf{x} \in \mathcal{X}^{\text{cand}}} \bar{\alpha}(\varphi_t(\mathbf{x}))$ ▷ Posterior and AF evaluated in a GPU batch

5 Experiments

We compare AF optimization with Adam (Kingma & Ba, 2014), L-BFGS (Liu & Nocedal, 1989) (default optimization method for BoTorch), FisherSAM (FSAM) (Kim et al., 2022), uniform random search (Uniform) Kim et al. (2025), and BOING (ours). Kim et al. (2025) show that uniform AF search can be sufficient for BO on low-dimensional problems. We test whether this holds on the higher-dimensional benchmarks as well. BoTorch (Balandat et al., 2020) was used for implementation.

Each run starts from 20 Sobol observations and evaluates one new input per BO iteration. We normalize all inputs to the unit cube and standardize outputs before training. If required, new input locations outside the unit cube are clipped to its bounds before evaluation. We use LogEI (Ament et al., 2023) and GIBBON (Moss et al., 2021), with 300 iterations for LunarLander (Eriksson et al., 2019), RobotPushing (Eriksson et al., 2019), Swimming (Wang et al., 2020), Hopper (Wang et al., 2020), and Rover (Eriksson et al., 2019); 750 for HalfCheetah (Wang et al., 2020), MOPTA08 (Eriksson et al., 2019), and LassoDNA (Šehić et al., 2022); and 1,000 for SVM (Eriksson & Poloczek, 2021; Papenmeier et al., 2022) and Ant (Wang et al., 2020). We run 15 independent trials per configuration. Adam, FSAM, and L-BFGS use ten restarts. They use the same starting-location distribution as BOING-H, based on BoTorch's default RAASP perturbation (Balandat et al., 2020), and select the starting locations from 512 initial candidates. Uniform uses $100(t+1)$ samples from $\mathcal{X}$, following Kim et al. (2025). We show two variants BOING-L and BOING-H. As explained in Appendix C.1, they differ only in the way the starting locations are created.

Discussion

Figures 4 and 5 show that BOING performs as well as, or in some benchmarks even better than, conventionally used iterative optimization methods.

Uniform search. The other non-iterative method, Uniform, is competitive in BO performance for lower-dimensional problems. However, it is suboptimal for high-dimensional settings such as MOPTA08, SVM, and Ant. The growing sample budget of $100 \times T$, as suggested by Kim et al. (2025), also increases time and memory requirements as input dimensions D_x increase.

Fisher SAM and radius selection. FSAM is also competitive on some benchmarks, but choosing its radius is non-trivial, and testing multiple radii adds computational overhead. Since BOING searches several radii in parallel, as shown in Algorithm 1, it is more efficient than FSAM. For GIBBON, BOING-H reaches much higher final values than Adam and L-BFGS on MOPTA08, SVM, and Ant.

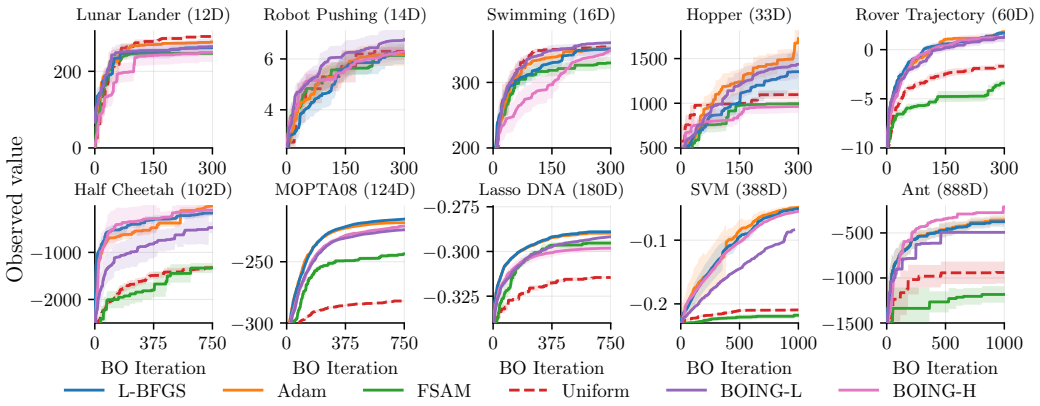


Figure 4: BO using LogEI (Ament et al., 2023) with mean and standard error. Higher is better.

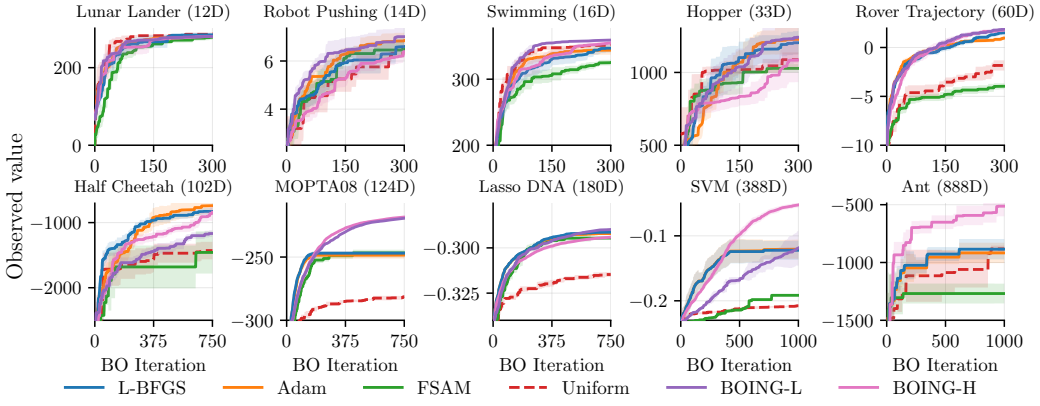


Figure 5: BO using Gibbon (Moss et al., 2021) with mean and standard error. Higher is better.

One possible explanation is that the starting locations and radius sweep give better coverage of $\mathcal{X}$, while the one-dimensional parallel search keeps the AF optimization cost small.

Starting locations and search choices. Figures 11 and 12 in Appendix C.4 compare the effect of the number of starting locations for BOING. We test two configurations: $R = t/2$ and $R = t/4$. The larger budget does not consistently yield better results, so we use $R = t/4$, as shown in Table 3.

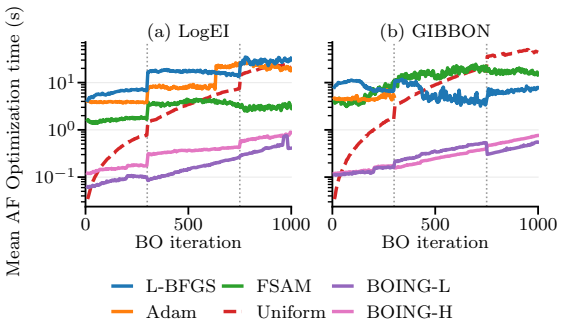


Figure 6: Mean AF optimization time over different benchmarks at each iteration. Per-benchmark times are shown in Figures 8 and 10.

Appendix C.1 explains that the starting locations consist of a mix of uniformly sampled and perturbed locations, as well as the incumbent location. Appendix C.5 shows the share of different sources of starting locations $\boldsymbol{x}_r$ from which the optimizer for a given BO iteration t is selected, averaged over multiple trials. Appendices C.6 and C.7 show the selected simplex coordinate ω and constraint radius multiplier c, averaged over the 15 trials. Results from Appendix C.7 support searching over starting sources, directions, and step sizes as explained in Figure 3.

Time and memory. The total time taken for BO primarily comprises model fitting, AF optimization, and benchmark evaluation. Since we only address the AF optimization, the other two stay the same for the respective BO experiments. Figure 6 compares AF optimization time averaged over different benchmarks and trials. Table 1 discusses it in detail. BOING and the iterative baselines share the starting location distribution. BOING saves computational effort by reusing the Jacobian from two batched backward passes for different weights and radii. Adam, L-BFGS, and FSAM repeatedly evaluate and differentiate the AF. To limit the growth for BOING, we cap the number of sampled starting locations at 256, in addition to the best observed input. Figures 7 and 9 in Appendix C.3 show total AF optimization time. GP fitting scales as $\mathcal{O}(T^3)$ (Rasmussen, 2003), so faster fitting, for example with a Stochastic Variational GP (Hensman et al., 2015; 2013), could make AF savings a larger share of the total. Tables 6 and 7 in Appendix C.3 show that BOING uses slightly more memory than the other iterative methods. However, the computational time benefit is multifold compared to the fractional increase in memory usage.

Table 1: Mean AF optimization time per BO iteration (seconds) / total BO time (minutes); lower is better. The final column gives the speedup factor of BOING over L-BFGS ($\times$, higher is better).

Problem	Adam	L-BFGS	FisherSAM	Uniform	BOING	BOING speedup over L-BFGS
			LogEI			
LunarLander	1.90 / 36	2.35 / 39	0.84 / 31	0.20 / 28	**0.09 / 27**	**26 / 1.4**
RobotPushing	1.91 / 42	3.08 / 48	0.83 / 37	0.22 / 34	**0.12 / 33**	**26 / 1.5**
Swimming	2.69 / 53	2.34 / 51	1.06 / 45	0.21 / 41	**0.13 / 40**	**18 / 1.3**
Hopper	3.97 / 89	4.95 / 94	1.15 / 75	0.25 / 71	**0.13 / 70**	**38 / 1.3**
Rover	6.64 / 113	3.75 / 99	1.03 / 85	0.30 / 81	**0.15 / 81**	**25 / 1.2**
HalfCheetah	6.32 / 885	6.31 / 885	3.17 / 846	1.21 / 822	**0.24 / 809**	**27 / 1.1**
MOPTA08	7.36 / 231	7.73 / 236	3.57 / 184	1.40 / 157	**0.23 / 142**	**34 / 1.7**
LassoDNA	17.94 / 523	15.39 / 491	4.95 / 360	3.80 / 346	**0.40 / 303**	**39 / 1.6**
SVM	18.91 / 391	14.31 / 315	4.87 / 157	7.95 / 209	**0.41 / 83**	**35 / 3.8**
Ant	83.93 / 1680	70.34 / 1453	1.80 / 311	9.33 / 436	**0.55 / 290**	**129 / 5.0**
			GIBBON			
LunarLander	4.68 / 53	3.34 / 47	7.57 / 68	0.42 / 32	**0.11 / 30**	**30 / 1.5**
RobotPushing	3.64 / 56	3.95 / 57	1.87 / 47	0.52 / 40	**0.13 / 38**	**32 / 1.5**
Swimming	4.12 / 66	3.70 / 64	8.64 / 89	0.42 / 47	**0.13 / 46**	**28 / 1.4**
Hopper	5.35 / 90	6.37 / 95	17.58 / 151	0.57 / 66	**0.12 / 64**	**51 / 1.5**
Rover	8.01 / 115	7.06 / 110	3.79 / 94	0.76 / 79	**0.14 / 76**	**50 / 1.5**
HalfCheetah	9.10 / 919	9.68 / 926	5.21 / 870	4.80 / 865	**0.25 / 808**	**39 / 1.2**
MOPTA08	2.32 / 188	4.03 / 210	65.37 / 976	5.33 / 226	**0.23 / 162**	**18 / 1.3**
LassoDNA	25.27 / 537	24.28 / 524	24.67 / 529	5.08 / 284	**0.16 / 223**	**156 / 2.4**
SVM	10.68 / 261	10.30 / 255	58.84 / 1064	17.25 / 371	**0.24 / 87**	**43 / 2.9**
Ant	49.95 / 976	48.14 / 945	2.78 / 189	20.09 / 478	**0.39 / 150**	**123 / 6.3**

6 Conclusion

Using insights from the surrogate-induced geometry, we propose BOING, an iteration-free acquisition-function (AF) optimization method that uses multiple local constrained optimizations. We show that the optimization in input space can be reduced to a one-dimensional grid search in predictive space. We empirically show substantial AF optimization speedups with competitive Bayesian Optimization (BO) performance and peak GPU memory of a similar order to the iterative baselines. We test over multiple benchmarks for two AFs. The present method, however, is restricted to scalar sequential BO. Future work could extend the method to other AFs like knowledge gradient or parallel BO.

AI Use Statement

We used generative AI tools only for writing assistance and polishing the manuscript for clarity and readability. We did not use generative AI tools to develop the method, formulate mathematical claims, produce proofs, design or run experiments, implement the method, analyze data, or interpret results. The authors reviewed and edited all AI-assisted text and take responsibility of this work.

REFERENCES

Shun-ichi Amari. *Information geometry and its applications*. Springer, 2016.

Shun-ichi Amari and Hiroshi Nagaoka. *Methods of information geometry*, volume 191. American Mathematical Soc., 2000.

Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for Bayesian optimization. *Advances in Neural Information Processing Systems*, 36:20577–20612, 2023.

Georgios Arvanitidis, Miguel González-Duque, Alison Pouplin, Dimitrios Kalatzis, and Soren Hauberg. Pulling back information geometry. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 4872–4894. PMLR, 2022. URL https://proceedings.mlr.press/v151/arvanitidis22b.html.

Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. BoTorch: A framework for efficient monte-carlo Bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 978-0-521-83378-3. URL https://web.stanford.edu/~boyd/cvxbook/.

Francesco Carciaghi, Simone Magistri, Pierluigi Mansueto, and Fabio Schoen. A bi-objective optimization based acquisition strategy for batch Bayesian global optimization. *Computational Optimization and Applications*, 92(1):81–123, 2025. doi:10.1007/s10589-025-00696-7. URL https://link.springer.com/article/10.1007/s10589-025-00696-7.

George De Ath, Richard M. Everson, Alma A. M. Rahat, and Jonathan E. Fieldsend. Greed is good: Exploration and exploitation trade-offs in bayesian optimisation. *ACM Transactions on Evolutionary Learning and Optimization*, 1(1):1–22, 2021. doi:10.1145/3425501.

Colin Doumont, Donney Fan, Natalie Maus, Jacob R Gardner, Henry Moss, and Geoff Pleiss. We still don't understand high-dimensional bayesian optimization. *arXiv preprint arXiv:2512.00170*, 2025.

David Eriksson and Matthias Poloczek. Scalable constrained bayesian optimization. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 730–738. PMLR, 2021. URL https://proceedings.mlr.press/v130/eriksson21a.html.

David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2023.

James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. In *Artificial Intelligence and Statistics*, pp. 351–360. PMLR, 2015.

Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla Bayesian optimization performs great in high dimensions. *arXiv preprint arXiv:2402.02229*, 2024.

Noémie Jaquier and Leonel Rozo. High-dimensional bayesian optimization via nested riemannian manifolds. *Advances in Neural Information Processing Systems*, 33:20939–20951, 2020.

Noémie Jaquier, Leonel Rozo, Sylvain Calinon, and Mathias Bürger. Bayesian optimization meets riemannian manifolds in robot learning. In *Conference on Robot Learning*, pp. 233–246. PMLR, 2020.

Noémie Jaquier, Viacheslav Borovitskiy, Andrei Smolensky, Alexander Terenin, Tamim Asfour, and Leonel Rozo. Geometry-aware bayesian optimization in robotics using riemannian matérn kernels. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 794–805. PMLR, 2022. URL https://proceedings.mlr.press/v164/jaquier22a.html.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

Hwanwoo Kim, Chong Liu, and Yuxin Chen. Bayesian optimization with inexact acquisition: is random grid search sufficient? *arXiv preprint arXiv:2506.11831*, 2025.

Minyoung Kim, Da Li, Shell X Hu, and Timothy Hospedales. Fisher SAM: Information geometry and sharpness aware minimisation. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 11148–11161. PMLR, 2022. URL https://proceedings.mlr.press/v162/kim22f.html.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Saksham Kiroriwal, Julius Pfrommer, and Jürgen Beyerer. Bayesian optimization with Fisher information geometry: Gradient bounds and trust-region methods. In *Advances in Neural Information Processing Systems*, 2026. Accepted for publication at NeurIPS 2026.

Rémi Lam, Matthias Poloczek, Peter Frazier, and Karen E Willcox. Advances in bayesian optimization with applications in aerospace engineering. In *2018 AIAA Non-Deterministic Approaches Conference*, pp. 1656, 2018.

John M Lee. *Introduction to Riemannian manifolds*, volume 2. Springer, 2018.

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020. URL https://www.jmlr.org/papers/v21/17-678.html.

Jonas Mockus. The application of bayesian methods for seeking the extremum. *Towards global optimization*, 2:117, 1998.

Henry B. Moss, David S. Leslie, Javier González, and Paul Rayson. GIBBON: General-purpose information-based Bayesian optimisation. *Journal of Machine Learning Research*, 22(235):1–49, 2021.

Henry B Moss, Sebastian W Ober, and Victor Picheny. Inducing point allocation for sparse gaussian processes in high-throughput bayesian optimisation. In *International Conference on Artificial Intelligence and Statistics*, pp. 5213–5230. PMLR, 2023.

Leonard Papenmeier, Luigi Nardi, and Matthias Poloczek. Increasing the scope as you learn: Adaptive Bayesian optimization in nested subspaces. *Advances in Neural Information Processing Systems*, 35:11586–11601, 2022.

Leonard Papenmeier, Matthias Poloczek, and Luigi Nardi. Understanding high-dimensional Bayesian optimization. *arXiv preprint arXiv:2502.09198*, 2025.

Federico Pavesi, Antonio Candelieri, and Noémie Jaquier. Information theoretic Bayesian optimization over the probability simplex. In *Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence*, volume 337 of *Proceedings of Machine Learning Research*, pp. 5356–5373. PMLR, 2026. URL https://proceedings.mlr.press/v337/pavesi26a.html.

PyTorch Contributors. CUDA semantics: Memory management. PyTorch 2.10 documentation, 2026. URL https://docs.pytorch.org/docs/2.10/notes/cuda.html. Accessed September 24, 2026.

Bahador Rashidi, Kerrick Johnstonbaugh, and Chao Gao. Cylindrical Thompson sampling for high-dimensional Bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, volume 238, pp. 3502–3510. PMLR, 2024.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*. Springer, 2003.

Rommel G. Regis and Christine A. Shoemaker. Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization. *Engineering Optimization*, 45(5):529–555, 2013.

Leonel Rozo, Miguel González-Duque, Noémie Jaquier, and Søren Hauberg. Riemann^2: Learning riemannian submanifolds from riemannian data. In *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pp. 3097–3105. PMLR, 2025. URL https://proceedings.mlr.press/v258/rozo25a.html.

Kenan Šehić, Alexandre Gramfort, Joseph Salmon, and Luigi Nardi. LassoBench: A high-dimensional hyperparameter optimization benchmark suite for lasso. In *Proceedings of the First International Conference on Automated Machine Learning*, volume 188 of *Proceedings of Machine Learning Research*, pp. 2/1–24. PMLR, 2022. URL https://proceedings.mlr.press/v188/sehic22a.html.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias W Seeger. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE transactions on information theory*, 58(5):3250–3265, 2012.

Alessandra Tosi, Søren Hauberg, Alfredo Vellido, and Neil D Lawrence. Metrics for probabilistic geometries. *arXiv preprint arXiv:1411.7432*, 2014.

Linnan Wang, Rodrigo Fonseca, and Yuandong Tian. Learning search space partition for black-box optimization using monte carlo tree search. In *Advances in Neural Information Processing Systems*, volume 33, pp. 19511–19522, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/e2ce14e81dba66dbff9cbc35ecfdb704-Abstract.html.

Thomas M Whitehead. From high dimensions to one: The exact acquisition frontier for spherical linear bayesian optimization, July 2026. URL https://doi.org/10.5281/zenodo.21721503.

James Wilson, Frank Hutter, and Marc Deisenroth. Maximizing acquisition functions for bayesian optimization. *Advances in neural information processing systems*, 31, 2018.

James T Wilson, Riccardo Moriconi, Frank Hutter, and Marc Peter Deisenroth. The reparameterization trick for acquisition functions. *arXiv preprint arXiv:1712.00424*, 2017.

Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. Hyperparameter optimization for machine learning models based on bayesian optimization. *Journal of Electronic Science and Technology*, 17(1):26–40, 2019.

Zhitong Xu, Haitao Wang, Jeff M Phillips, and Shandian Zhe. Standard Gaussian process is all you need for high-dimensional Bayesian optimization. *arXiv preprint arXiv:2402.02746*, 2024.

Rosen Ting-Ying Yu, Cyril Picard, and Faez Ahmed. GIT-BO: High-dimensional Bayesian optimization with tabular foundation models. *arXiv preprint arXiv:2505.20685*, 2025.

A GAUSSIAN PROCESS

A Gaussian process (GP) is a distribution over functions, where we place a prior on the unknown objective and update our belief using the noisy observations Rasmussen (2003).

We model the latent function as

$$f \sim \mathcal{GP}\big(m(\cdot), k(\cdot, \cdot')\big),$$

where $m(\cdot)$ is the prior mean and $k(\cdot, \cdot')$ is the covariance kernel. For observed input-output pairs $\mathcal{D}_t = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$, we use the Gaussian observation model

$$y_t = f(\mathbf{x}_t) + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma_\varepsilon^2).$$

Given $\mathcal{D}_t$, the latent predictive distribution at an input location $\mathbf{x}$ is Gaussian

$$p(f(\mathbf{x}) \mid \mathcal{D}_t) = \mathcal{N}(\mu_t(\mathbf{x}),\, \Sigma_t(\mathbf{x}))\,,$$

where $\mathbf{X}$ and $\mathbf{y}$ collect the observed inputs and outputs. The latent posterior mean and variance follow from Gaussian conditioning Rasmussen (2003)

$$\mu_t(\mathbf{x}) = m(\mathbf{x}) + k(\mathbf{x}, \mathbf{X}) \left(k(\mathbf{X}, \mathbf{X}) + \sigma_\varepsilon^2 \mathbf{I}\right)^{-1} (\mathbf{y} - m(\mathbf{X}))\,, \tag{13}$$

$$\Sigma_t(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - k(\mathbf{x}, \mathbf{X}) \left(k(\mathbf{X}, \mathbf{X}) + \sigma_\varepsilon^2 \mathbf{I}\right)^{-1} k(\mathbf{X}, \mathbf{x}). \tag{14}$$

The variance of a future noisy observation is $\Sigma_t(\mathbf{x}) + \sigma_\varepsilon^2$.

B ACQUISITION FUNCTIONS AND OPTIMIZERS

Table 2 shows some closed-form AF for a scalar Gaussian predictive, where $z = (\mu_{t,\mathbf{x}} - y^\star)/\Sigma_{t,\mathbf{x}}^{1/2}$, $\mathrm{CDF}_{\mathcal{N}}$ is the Cumulative Distribution Function (CDF) and $\mathrm{PDF}_{\mathcal{N}}$ is Probability Density Function (PDF) of normal distribution. For sequential GIBBON, the posterior maximum-value samples and homoscedastic observation-noise variance are fixed during AF optimization. Here $y^\star$ is the best observed output and $\kappa_{\mathrm{UCB}} > 0$ is the UCB exploration weight. LogEI evaluates the logarithm of EI using the stable formulation of Ament et al. (2023).

Table 2: Analytic acquisitions for a scalar Gaussian predictive (Jones et al., 1998; Mockus, 1998; Srinivas et al., 2012).

Acquisition	**Closed-form** $\bar{\alpha}(\mu_{t,\mathbf{x}}, \Sigma_{t,\mathbf{x}}; \boldsymbol{\xi})$
EI	$\left(\mu_{t,\mathbf{x}} - y^\star\right) \mathrm{CDF}_{\mathcal{N}}(z) + \Sigma_{t,\mathbf{x}}^{1/2} \mathrm{PDF}_{\mathcal{N}}(z)$
PI	$\mathrm{CDF}_{\mathcal{N}}(z)$
UCB	$\mu_{t,\mathbf{x}} + \sqrt{\kappa_{\mathrm{UCB}}}\, \Sigma_{t,\mathbf{x}}^{1/2}$

B.1 SEQUENTIAL BO WITH A CHOICE OF AF OPTIMIZER

Algorithm 2 gives the sequential BO loop (Garnett, 2023; Frazier, 2018). The AF optimizer $\mathrm{AFOptimize}$ is any suitable optimization algorithm such as L-BFGS, or Adam (Kingma & Ba, 2014) or Fisher-SAM (Kim et al., 2022) as mentioned in Algorithm 3, or our method BOING (Algorithm 1).

Algorithm 2: Sequential BO with a selectable AF optimizer

Input: Objective f_{true}; domain $\mathcal{X}$; initial data $\mathcal{D}_0$; BO budget T; AF $\bar{\alpha}$; Optimization method AFOptimize

Output: Best observed output $y^\star$

for $t = 0, \ldots, T-1$ **do**

  Fit surrogate $\mathcal{M}_t$ using $\mathcal{D}_t$

  $\mathbf{x}_{t+1} \leftarrow \text{AFOptimize}(\bar{\alpha}(\cdot; \boldsymbol{\xi}_t), \mathcal{M}_t, \mathcal{X})$ ▷ which approximately solves $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x} \in \mathcal{X}} \bar{\alpha}(\varphi_t(\mathbf{x}); \boldsymbol{\xi}_t)$

  $y_{t+1} \leftarrow f_{\text{true}}(\mathbf{x}_{t+1})$

  $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(\mathbf{x}_{t+1}, y_{t+1})\}$

return $y^\star = \max\{y : (\mathbf{x}, y) \in \mathcal{D}_T\}$

B.2 Fisher-SAM Acquisition Optimization

Fisher SAM (Kim et al., 2022) and BOING (our method) both use a local Fisher ellipsoid to constrain the update of the objective function decision variable, $\mathbf{x}$ for Fisher SAM and $\boldsymbol{\theta}$ for BOING. However as an approximation, Fisher SAM uses first-order approximation of the objective function to obtain an optimal update direction (Kim et al., 2022, Eqs. 8–10). The iterative optimization process is repeated until the stopping criteria is met (Kim et al., 2022, Alg. 1).

BOING shows that each local non-convex optimization under the stated assumptions of AF on the manifold can be reduced to a 1-D simplex search, which we approximate with a finite sweep. Also, BOING does not require sequential gradient steps for AF optimization, and its candidate scores can be evaluated in parallel using batched computation on GPU.

Algorithm 3 adapts Fisher SAM (Kim et al., 2022) to input-space AF optimization using the loss $-\alpha$ and a fixed fitted surrogate. At each BO iteration t, it runs an iterative optimization of the AF until the inner optimization budget I is finished or a stopping criteria is met. Here diag means the diagonal entries retained as a diagonal matrix and ε is small floor value to avoid errors in matrix inverse. GradientUpdate means the gradient-ascent update routine like in Adam or SGD or momentum based gradient update Kingma & Ba (2014). In the implementation, the perturbation is held fixed for the outer gradient and proposed inputs are projected to $\mathcal{X}$.

Algorithm 3: Fisher-SAM-inspired AF optimization at BO iteration t from R starting locations

Input: Surrogate $\mathcal{M}_t$, AF α; Input space $\mathcal{X}$; starting locations $\{\mathbf{x}_r^{\text{start}}\}_{r=1}^{R} \subseteq \mathcal{X}$; radius $\kappa \geq 0$; optimizer budget I per starting location; GradientUpdate routine

Output: Next input location $\mathbf{x}_{t+1}$

$\mathcal{X}_t^{\text{cand}} \leftarrow \varnothing$

for $r \in 1, \ldots, R$ **do**

  $\mathbf{x} \leftarrow \mathbf{x}_r^{\text{start}}$; reset the GradientUpdate state

  $i \leftarrow 0$

  while $i < I$ *and stopping criteria is not met* **do**

    Evaluate the diagonal and trace of $\mathbf{g}_t(\mathbf{x})$ using (2)

    $\mathbf{g}_t(\mathbf{x}) \leftarrow \text{diag}(\mathbf{g}_t(\mathbf{x})) + \left(\dfrac{\text{tr}(\mathbf{g}_t(\mathbf{x}))}{D_x} + \varepsilon\right)\mathbf{I}$

    $\boldsymbol{d} \leftarrow \mathbf{0}$

    if $\kappa > 0$ *and* $\nabla_{\mathbf{x}}\alpha(\mathbf{x}) \neq \mathbf{0}$ **then**

      $\boldsymbol{d} \leftarrow -\dfrac{\kappa\,[\mathbf{g}_t(\mathbf{x})]^{-1}\nabla_{\mathbf{x}}\alpha(\mathbf{x})}{\sqrt{[\nabla_{\mathbf{x}}\alpha(\mathbf{x})]^\top [\mathbf{g}_t(\mathbf{x})]^{-1}\nabla_{\mathbf{x}}\alpha(\mathbf{x})}}$

    $\mathbf{x}^{\text{pert}} \leftarrow \mathbf{x} + \boldsymbol{d}$

    $\mathbf{x} \leftarrow \text{GradientUpdate}(\mathbf{x}, \nabla_{\mathbf{x}}\alpha(\mathbf{x}^{\text{pert}}))$

    $i \leftarrow i + 1$

  $\mathcal{X}_t^{\text{cand}} \leftarrow \mathcal{X}_t^{\text{cand}} \cup \{\mathbf{x}\}$

return $\arg\max_{\mathbf{x} \in \mathcal{X}_t^{\text{cand}}} \alpha(\mathbf{x})$

C Implementation Details

C.1 BOING-L and BOING-H

The starting locations in Algorithm 1 combine perturbations around so far top performing observed inputs and uniformly sampled inputs. We propose BOING-L and BOING-H, which differ in the way perturbed starting locations as created. We use a $50\% - 50\%$ mix of the pertrubed location and uniformly sampled locations from $\mathcal{X}$, as proposed by RAASP (Papenmeier et al., 2025; Eriksson et al., 2019). The leading 1 in the starting-location count in Table 3 is the best observed input so far.

BOING-H uses BoTorch's default RAASP perturbation (Balandat et al., 2020). Adam, FSAM, and L-BFGS draw initial candidates from this same starting-location distribution and select their restart points from that pool. However we propose a different perturbation configuration for BOING-L since the perturbations used in BOING-H are very small, it can lead to suboptimal behaviour in low input dimensions D_x. The implementation uses N_ω uniformly spaced coefficients $\omega \in [0, 1]$ to approximate the local search in (11); numerical coordinate details are given in Appendix C.2.

Table 3: Starting-location and search settings for BOING-L and BOING-H. Each radius is the multiplier times the mean fitted lengthscale, as in (15).

Setting	BOING-L	BOING-H
Number of starting locations R at BO iteration t	$1 + \min(256, \max(16, \lceil \lvert\mathcal{D}_t\rvert/4 \rceil))$	$1 + \min(256, \max(16, \lceil \lvert\mathcal{D}_t\rvert/4 \rceil))$
Pertrubed to uniform mixing	$50\% - 50\%$	$50\% - 50\%$
Top performing input locations	Best 20%	Best 5%
Coordinate perturbation rule	20% of D_x	$\min(20/D_x, 1)$
Perturbation scale	$0.5\lambda_j$ (lengthscale of the selected dimension)	0.001
Fisher radius multipliers $c \in \mathcal{C}_{\text{grid}}$	$(0.05,\ 0.1,\ 0.2,\ 0.3,$ $0.4,\ 0.5,\ 0.8,\ 1.0)$	$(0.05,\ 0.1,\ 0.2,\ 0.3,$ $0.4,\ 0.5,\ 0.8,\ 1.0)$
Number of search points on 1-D simplex N_ω	32	32
Maximum batch size for posterior evaluation	512	512

C.2 BOING Algorithm for a Univariate GP: Implementation

Algorithm 1 uses the finite grids $\mathcal{C}_{\text{grid}}$ and $\mathcal{W}_{\text{grid}}$ given in Table 3. For each radius multiplier c, the Fisher radius is

$$\kappa_t(c) := c\bar{\lambda}_t, \qquad c \in \mathcal{C}_{\text{grid}}. \tag{15}$$

For the univariate Gaussian predictive parameter $\boldsymbol{\theta}_r = (\mu_r, \Sigma_r)^\top$, the matrix multiplication in (12) is

$$\mathcal{I}_r^{-1}\boldsymbol{\Omega} = \begin{bmatrix} \Sigma_r & 0 \\ 0 & 2\Sigma_r^2 \end{bmatrix} \begin{bmatrix} \omega \\ 1-\omega \end{bmatrix} = \begin{bmatrix} \Sigma_r\omega \\ 2\Sigma_r^2(1-\omega) \end{bmatrix}. \tag{16}$$

Substituting (16) into (12) gives the predictive update in variance coordinates,

$$\boldsymbol{\Delta\theta}_r(\kappa, \omega) = \frac{\kappa}{\sqrt{\Sigma_r\omega^2 + 2\Sigma_r^2(1-\omega)^2}} \begin{bmatrix} \Sigma_r\omega \\ 2\Sigma_r^2(1-\omega) \end{bmatrix}. \tag{17}$$

The update uses only Σ_r, κ, and ω, without forming a Fisher matrix. Each ω then directly gives the updated predictive parameter

$$\boldsymbol{\theta}_r(\kappa_t(c), \omega) = \boldsymbol{\theta}_r + \boldsymbol{\Delta\theta}_r(\kappa_t(c), \omega)\,, \tag{18}$$

which is (12) for univariate GP. We evaluate $\bar{\alpha}$ directly at the N_ω predictive parameters and (11) can then be approximated by a grid search as

$$\omega^\star_{r,c} \in \operatorname*{arg\,max}_{\omega \in \mathcal{W}_{\text{grid}}} \bar{\alpha}(\boldsymbol{\theta}_r(\kappa_t(c), \omega))\,. \tag{19}$$

This evaluates the AF on the two-dimensional predictive manifold using grid search on 1-D line and does not require a GP posterior evaluation for every grid point. Hence reducing the optimizaion of AF in D_x space to 1-D space.

In this formulation, only the best sampled predictive update at each radius from each location r is mapped back to the input space and the AF is evaluated using the expensive model posterior. Under Assumption 4.3, the pseudoinverse used in (6) is

$$J_r^\dagger = J_r^\top (J_r J_r^\top)^{-1},$$
$$J_r J_r^\top = \begin{bmatrix} \|\nabla_{\mathbf{x}} \mu_r\|_2^2 & \nabla_{\mathbf{x}} \mu_r^\top \nabla_{\mathbf{x}} \Sigma_r \\ \nabla_{\mathbf{x}} \Sigma_r^\top \nabla_{\mathbf{x}} \mu_r & \|\nabla_{\mathbf{x}} \Sigma_r\|_2^2 \end{bmatrix}. \tag{20}$$

Using this expanded form in (6), the winning input displacement and candidate are

$$\boldsymbol{d}_{r,c}^\star := J_r^\dagger \boldsymbol{\Delta\theta}_r\big(\kappa_t(c), \omega_{r,c}^\star\big),$$
$$\mathbf{x}_{r,c}^\star := \mathbf{x}_r + \boldsymbol{d}_{r,c}^\star. \tag{21}$$

The input displacement is therefore obtained from the maximizing predictive update. The candidate set in (7) becomes

$$\mathcal{X}^{\text{cand}} := \{\mathbf{x}_r\}_{r=1}^R \cup \Big(\bigcup_{r=1}^R \{\mathbf{x}_{r,c}^\star \; \forall \; c \in \mathcal{C}_{\text{grid}}\} \Big). \tag{22}$$

The GP posterior is evaluated at $\mathcal{X}^{\text{cand}}$ in GPU batches, and the next input is selected using the final argmax in (7).

The pseudoinverse gives a local first-order realization of the selected predictive update. Since $J_r J_r^\dagger = \mathbf{I}_2$ under Assumption 4.3,

$$\varphi_t\big(\mathbf{x}_r + \boldsymbol{d}_{r,c}^\star\big) = \boldsymbol{\theta}_r + \boldsymbol{\Delta\theta}_r\big(\kappa_t(c), \omega_{r,c}^\star\big) + o\big(\|\boldsymbol{d}_{r,c}^\star\|_2\big). \tag{23}$$

The exact posterior evaluation in (7) scores the retained candidates using the true GP posterior; it does not evaluate the discarded candidates. If the input displacement is outside the bounds of $\mathcal{X}$, the respective dimensions are clipped to make it within bounds. The final posterior evaluation scores only the retained candidates and original starts in (22); however the discared candidates from each local search are discarded. Since we use Jacobian to map the predictive parameters back to the input space, this leads to a first order approximation. Figure 3 illustrates this effect without clipping: a larger predicted AF need not give a larger AF at the mapped input using true posterior map φ_t.

C.3 Time and Memory Usage Comparison

Timing results are reported over 15 independent trials per configuration. Table 4 gives the mean and sample standard deviation of the computation time for AF optimization for one BO iteration. Table 5 adds the same minimum model-fitting and benchmark-evaluation time for each problem and AF to the recorded AF time of every run. The timing tables use BOING-L for the first five problems and BOING-H for the others, reported under BOING.

Memory usage. Tables 6 and 7 report the mean and sample standard deviation of each run's maximum recorded BO-iteration peak. Counters reset before GP fitting and are read after objective evaluation. Allocated memory covers live PyTorch tensors; reserved memory also includes reusable allocator blocks (PyTorch Contributors, 2026). These process counters exclude other jobs' allocations. Their measurement window spans all BO phases rather than AF optimization alone. We use the same BOING variants as in the timing tables.

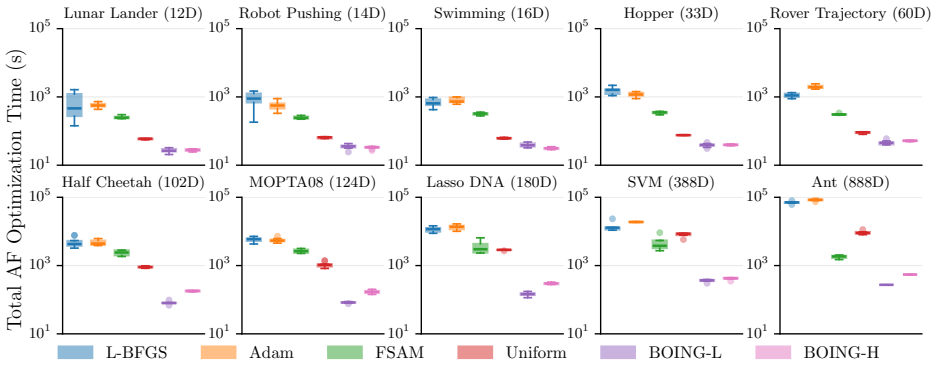


Figure 7: Total AF optimization time per BO run with LogEI.

Table 4: AF optimization time in seconds per BO iteration, reported as mean and sample standard deviation across runs.

Problem	Adam	L-BFGS	FisherSAM	Uniform random	BOING
			LogEI		
LunarLander	1.896 ± 0.271	2.353 ± 1.711	0.843 ± 0.076	0.195 ± 0.007	$\mathbf{0.091 \pm 0.011}$
RobotPushing	1.908 ± 0.525	3.077 ± 1.347	0.831 ± 0.076	0.215 ± 0.012	$\mathbf{0.119 \pm 0.015}$
Swimming	2.693 ± 0.461	2.341 ± 0.559	1.063 ± 0.080	0.206 ± 0.009	$\mathbf{0.132 \pm 0.017}$
Hopper	3.968 ± 0.507	4.948 ± 1.016	1.151 ± 0.089	0.254 ± 0.004	$\mathbf{0.129 \pm 0.013}$
Rover	6.643 ± 0.791	3.747 ± 0.497	1.027 ± 0.043	0.302 ± 0.015	$\mathbf{0.150 \pm 0.019}$
HalfCheetah	6.322 ± 1.054	6.312 ± 1.828	3.173 ± 0.679	1.214 ± 0.058	$\mathbf{0.238 \pm 0.010}$
MOPTA08	7.355 ± 0.831	7.730 ± 1.105	3.574 ± 0.558	1.404 ± 0.228	$\mathbf{0.227 \pm 0.028}$
LassoDNA	17.937 ± 3.131	15.389 ± 2.701	4.948 ± 2.592	3.803 ± 0.110	$\mathbf{0.398 \pm 0.021}$
SVM	18.913 ± 0.442	14.314 ± 5.181	4.873 ± 2.929	7.945 ± 1.450	$\mathbf{0.414 \pm 0.038}$
Ant	83.931 ± 7.250	70.337 ± 6.500	1.796 ± 0.273	9.334 ± 1.414	$\mathbf{0.545 \pm 0.326}$
			GIBBON		
LunarLander	4.684 ± 5.085	3.338 ± 1.045	7.574 ± 8.342	0.418 ± 0.023	$\mathbf{0.113 \pm 0.018}$
RobotPushing	3.639 ± 1.506	3.952 ± 1.074	1.868 ± 0.552	0.524 ± 0.016	$\mathbf{0.125 \pm 0.028}$
Swimming	4.120 ± 0.531	3.695 ± 0.914	8.636 ± 5.967	0.422 ± 0.018	$\mathbf{0.134 \pm 0.014}$
Hopper	5.345 ± 1.615	6.373 ± 2.905	17.577 ± 14.438	0.572 ± 0.021	$\mathbf{0.124 \pm 0.022}$
Rover	8.010 ± 0.860	7.064 ± 0.794	3.789 ± 1.691	0.756 ± 0.027	$\mathbf{0.141 \pm 0.030}$
HalfCheetah	9.103 ± 1.241	9.680 ± 0.775	5.205 ± 0.446	4.795 ± 0.042	$\mathbf{0.249 \pm 0.007}$
MOPTA08	2.318 ± 0.424	4.028 ± 0.729	65.369 ± 1.857	5.326 ± 0.074	$\mathbf{0.228 \pm 0.017}$
LassoDNA	25.273 ± 8.236	24.280 ± 7.963	24.671 ± 3.972	5.079 ± 0.150	$\mathbf{0.156 \pm 0.039}$
SVM	10.679 ± 2.987	10.301 ± 2.885	58.840 ± 4.78	17.253 ± 0.145	$\mathbf{0.237 \pm 0.004}$
Ant	49.954 ± 3.344	48.139 ± 2.545	2.776 ± 0.421	20.089 ± 0.841	$\mathbf{0.390 \pm 0.013}$

Table 5: Total BO time in minutes. Budgets are 300 iterations for the first five problems, 750 for HalfCheetah, MOPTA08, and LassoDNA, and 1,000 for SVM and Ant.

Problem	Adam	L-BFGS	FisherSAM	Uniform random	BOING
			LogEI		
LunarLander	36.3 ± 1.4	38.6 ± 8.6	31.0 ± 0.4	27.8 ± 0.0	$\mathbf{27.3 \pm 0.1}$
RobotPushing	42.2 ± 2.6	48.0 ± 6.7	36.8 ± 0.4	33.7 ± 0.1	$\mathbf{33.2 \pm 0.1}$
Swimming	53.1 ± 2.3	51.4 ± 2.8	45.0 ± 0.4	40.7 ± 0.0	$\mathbf{40.3 \pm 0.1}$
Hopper	89.1 ± 2.5	94.0 ± 5.1	75.0 ± 0.4	70.5 ± 0.0	$\mathbf{69.9 \pm 0.1}$
Rover	113.0 ± 4.0	98.5 ± 2.5	84.9 ± 0.2	81.3 ± 0.1	$\mathbf{80.5 \pm 0.1}$
HalfCheetah	885.4 ± 13.2	885.2 ± 22.8	846.0 ± 8.5	821.5 ± 0.7	$\mathbf{809.3 \pm 0.1}$
MOPTA08	231.2 ± 10.4	235.9 ± 13.8	183.9 ± 7.0	156.8 ± 2.9	$\mathbf{142.1 \pm 0.3}$
LassoDNA	522.5 ± 39.1	490.6 ± 33.8	360.1 ± 32.4	345.8 ± 1.4	$\mathbf{303.2 \pm 0.3}$
SVM	391.4 ± 7.4	314.8 ± 86.3	157.4 ± 48.8	208.6 ± 24.2	$\mathbf{83.1 \pm 0.6}$
Ant	1679.5 ± 120.8	1452.9 ± 108.3	310.6 ± 4.5	436.2 ± 23.6	$\mathbf{289.7 \pm 0.5}$
			GIBBON		
LunarLander	53.2 ± 25.4	46.5 ± 5.2	67.6 ± 41.7	31.9 ± 0.1	$\mathbf{30.3 \pm 0.1}$
RobotPushing	55.8 ± 7.5	57.3 ± 5.4	46.9 ± 2.8	40.2 ± 0.1	$\mathbf{38.2 \pm 0.1}$
Swimming	65.9 ± 2.7	63.8 ± 4.6	88.5 ± 29.8	47.4 ± 0.1	$\mathbf{46.0 \pm 0.1}$
Hopper	89.8 ± 8.1	94.9 ± 14.5	151.0 ± 72.2	65.9 ± 0.1	$\mathbf{63.7 \pm 0.1}$
Rover	115.1 ± 4.3	110.4 ± 4.0	94.0 ± 8.5	78.8 ± 0.1	$\mathbf{75.8 \pm 0.1}$
HalfCheetah	918.9 ± 15.5	926.1 ± 9.7	870.2 ± 5.6	865.0 ± 0.5	$\mathbf{808.2 \pm 0.1}$
MOPTA08	188.3 ± 5.3	209.7 ± 9.1	976.4 ± 23.2	225.9 ± 0.9	$\mathbf{162.2 \pm 0.2}$
LassoDNA	536.6 ± 102.9	524.2 ± 99.5	529.1 ± 49.6	284.2 ± 1.9	$\mathbf{222.6 \pm 0.5}$
SVM	261.2 ± 49.8	254.9 ± 48.1	1063.9 ± 47.7	370.8 ± 2.4	$\mathbf{87.2 \pm 0.1}$
Ant	975.6 ± 55.7	945.3 ± 42.4	189.3 ± 7.0	477.8 ± 14.0	$\mathbf{149.5 \pm 0.2}$

Table 6: Peak allocated GPU memory (MiB), mean $\pm$ sample standard deviation of run-level peaks.

Problem	Adam	L-BFGS	FisherSAM	BOING
		LogEI		
LunarLander	142 ± 1	142 ± 0	143 ± 1	220 ± 0
RobotPushing	154 ± 1	154 ± 1	155 ± 1	231 ± 1
Swimming	162 ± 0	162 ± 0	163 ± 1	242 ± 1
Hopper	250 ± 1	251 ± 3	250 ± 1	341 ± 2
Rover	388 ± 1	389 ± 2	388 ± 1	498 ± 1
HalfCheetah	1615 ± 11	1615 ± 11	2005 ± 14	1905 ± 7
MOPTA08	1866 ± 9	1866 ± 9	2278 ± 9	2154 ± 10
LassoDNA	2781 ± 347	2784 ± 348	2966 ± 4	3059 ± 164
SVM	8080 ± 26	7709 ± 539	8100 ± 23	8313 ± 255
Ant	15483 ± 362	15499 ± 362	15060 ± 681	16324 ± 782
		GIBBON		
LunarLander	149 ± 0	149 ± 1	149 ± 0	172 ± 1
RobotPushing	161 ± 0	160 ± 1	160 ± 0	184 ± 0
Swimming	170 ± 0	170 ± 0	170 ± 1	193 ± 0
Hopper	254 ± 0	255 ± 2	255 ± 1	285 ± 1
Rover	390 ± 1	397 ± 3	390 ± 0	433 ± 1
HalfCheetah	2409 ± 260	2409 ± 260	2029 ± 16	2052 ± 4
MOPTA08	2251 ± 15	2251 ± 15	2319 ± 3	2326 ± 13
LassoDNA	2752 ± 162	2751 ± 162	2987 ± 5	2930 ± 9
SVM	7840 ± 430	7837 ± 430	8112 ± 320	8303 ± 20
Ant	16601 ± 704	16596 ± 704	15247 ± 306	18494 ± 580

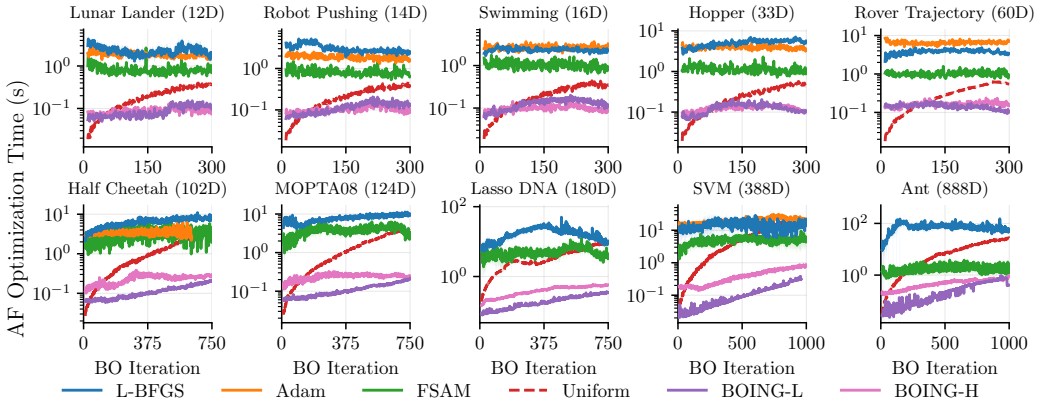


Figure 8: LogEI AF optimization time (seconds) during BO for different benchmarks. Lower is better.

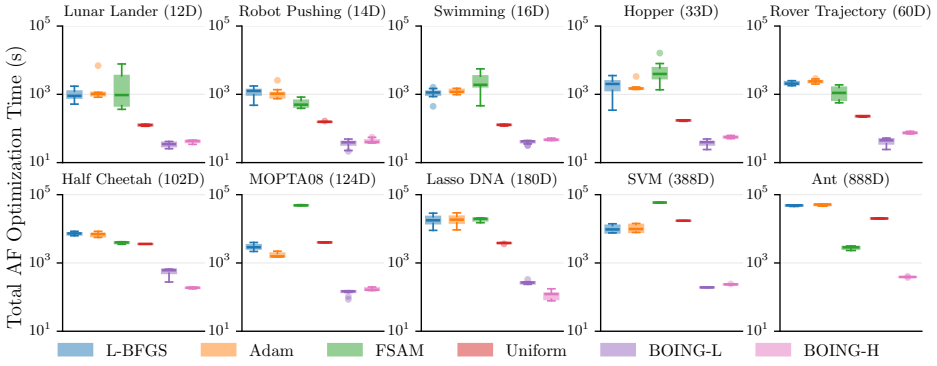


Figure 9: Total AF optimization time per BO run with GIBBON.

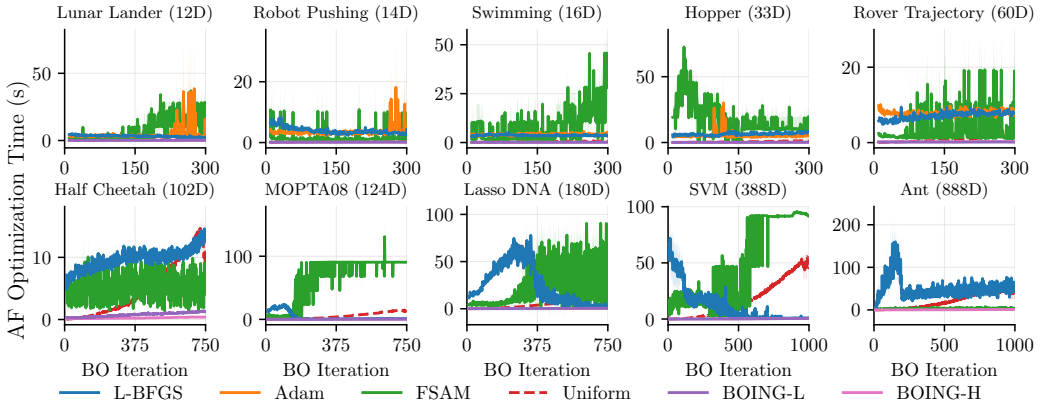


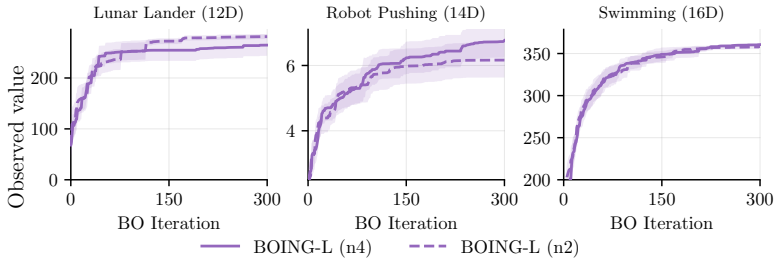
Figure 10: Gibbon AF optimization time (seconds) during BO for different benchmarks. Lower is better.

Table 7: Peak reserved GPU memory (MiB), mean ± sample standard deviation of run-level peaks.

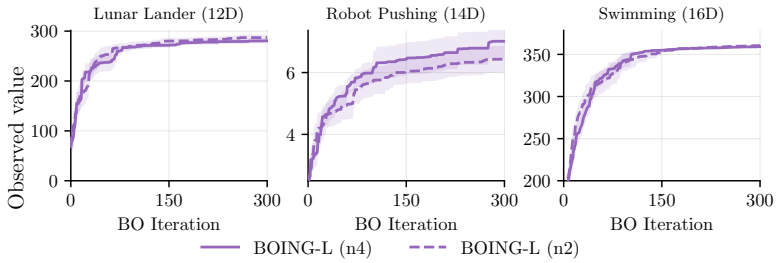
Problem	Adam	L-BFGS	FisherSAM	BOING
		LogEI		
LunarLander	210 ± 2	210 ± 2	211 ± 2	284 ± 0
RobotPushing	220 ± 3	219 ± 1	220 ± 3	294 ± 2
Swimming	231 ± 2	232 ± 2	233 ± 3	326 ± 0
Hopper	341 ± 2	340 ± 2	343 ± 1	434 ± 1
Rover	507 ± 2	509 ± 3	515 ± 2	635 ± 2
HalfCheetah	2199 ± 14	2199 ± 14	2722 ± 32	2628 ± 26
MOPTA08	2524 ± 15	2524 ± 15	3090 ± 22	3011 ± 98
LassoDNA	3719 ± 496	3718 ± 496	3918 ± 7	4076 ± 216
SVM	10758 ± 300	10356 ± 543	10431 ± 22	10952 ± 298
Ant	19880 ± 880	19874 ± 880	16599 ± 840	20682 ± 854
		GIBBON		
LunarLander	243 ± 2	242 ± 2	242 ± 2	258 ± 2
RobotPushing	251 ± 2	251 ± 2	251 ± 3	268 ± 2
Swimming	261 ± 2	261 ± 2	261 ± 3	276 ± 1
Hopper	390 ± 2	389 ± 2	390 ± 2	405 ± 1
Rover	552 ± 2	562 ± 6	554 ± 2	578 ± 3
HalfCheetah	3393 ± 342	3393 ± 342	2805 ± 9	2774 ± 14
MOPTA08	3085 ± 14	3085 ± 14	3187 ± 21	3106 ± 25
LassoDNA	3683 ± 237	3677 ± 236	4011 ± 44	3831 ± 19
SVM	10190 ± 270	10172 ± 269	10508 ± 120	10579 ± 8
Ant	20840 ± 749	20805 ± 748	16836 ± 401	20708 ± 355

C.4 Effect of the Number of Starting Locations

Figures 11 and 12 compare the number of starting locations for BOING-L and BOING-H. We compare two configurations: $t/4$ (n4) and $t/2$ (n2), respectively. We observed that the optimization results differ initially for the two configuration, but the final observed values are broadly similar. Differences remain on Robot Pushing and Hopper, with a larger difference for BOING-H on Hopper with GIBBON. However, the larger starting-location budget does not give a consistent improvement across problems and AFs. We choose n4 in our experiments in Section 5, since it gives comparable performance in most cases with fewer local searches. The displayed comparison therefore supports the starting-location budget used in the main experiments.

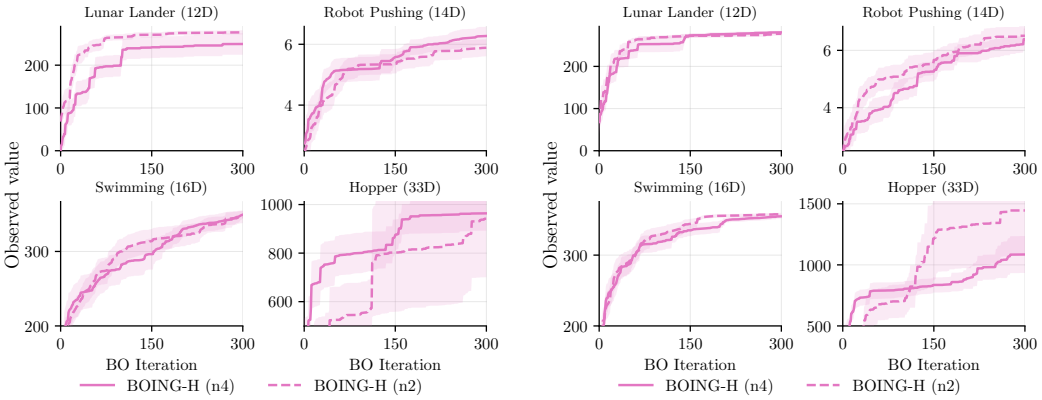


(a) LogEI



(b) GIBBON

Figure 11: Starting-location comparison for BOING-L; higher is better.



(a) LogEI

(b) GIBBON

Figure 12: Comparison of number of starting locations for BOING-H.

C.5 Selected Starting Locations

Figures 13–16 show the fraction of times the final selected candidate came from the best observed input, a uniformly sampled location in $\mathcal{X}$, or a perturbed starting location; averaged over the 15 independent trails. Here optimal refers to the highest AF value among the finite candidates evaluated in Algorithm 1.

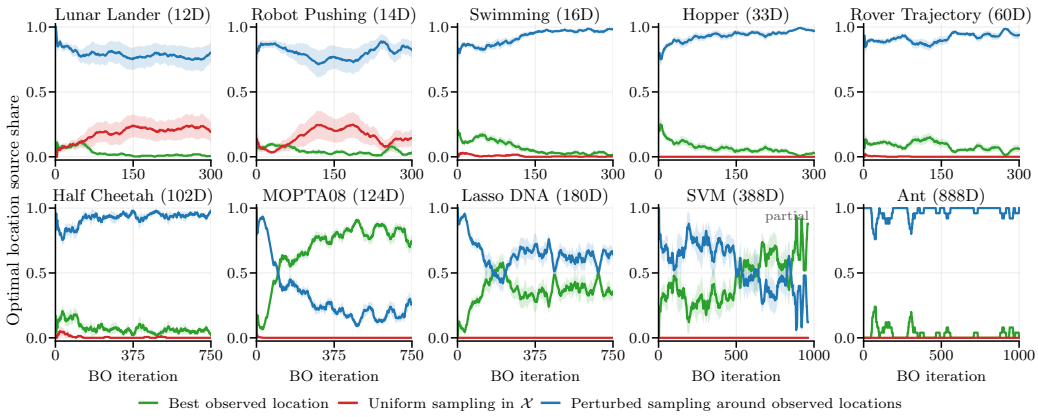


Figure 13: Optimal location source share for BOING-L with LogEI.

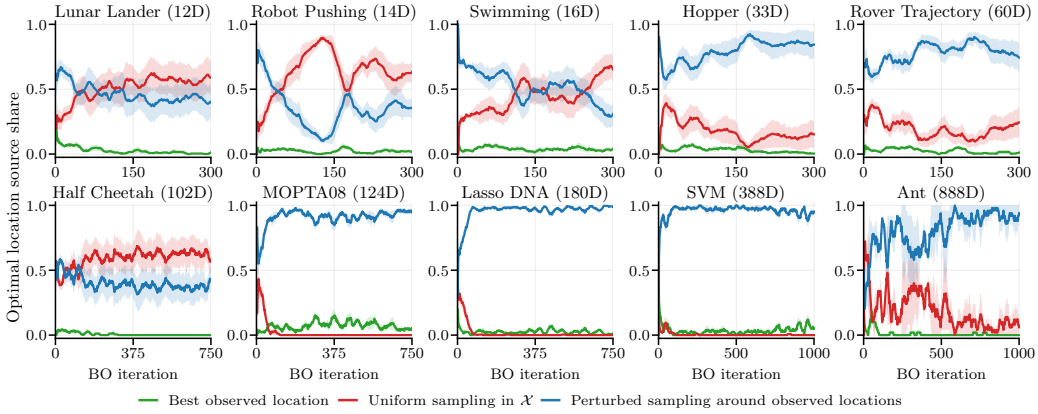


Figure 14: Optimal location source share for BOING-H with LogEI.

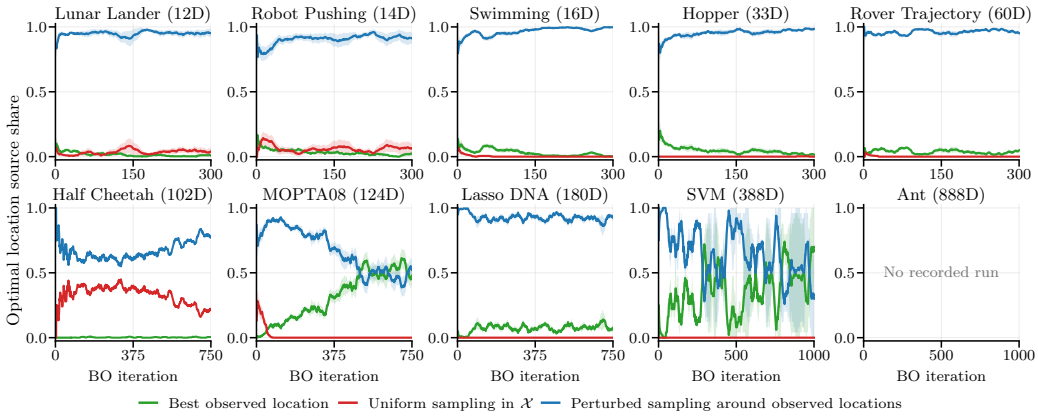


Figure 15: Optimal location source share for BOING-L with GIBBON.

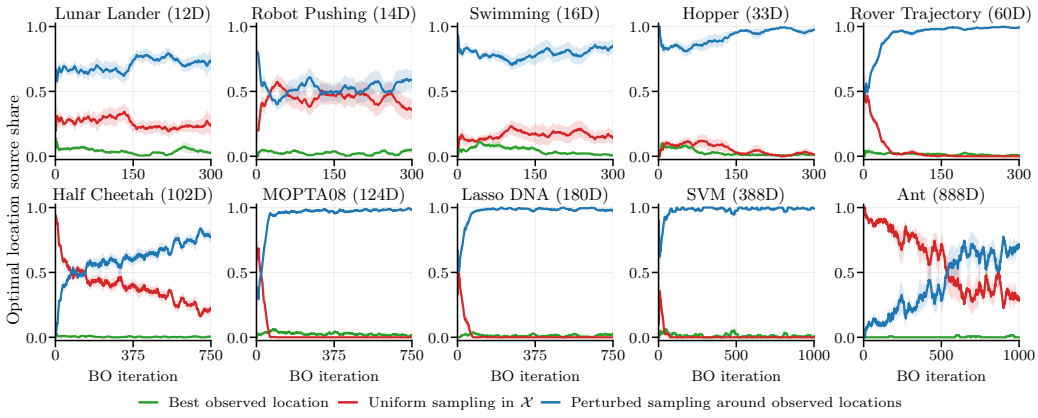


Figure 16: Optimal location source share for BOING-H with GIBBON.

C.6 Optimal 1-D Simplex Search Weight

The plots denote the selected simplex grid coordinate by ω_{grid}. Larger values of ω give more weight to the mean direction. Figures 17–20 shows the mean across runs. The different selected balances motivate searching over weights across these settings.

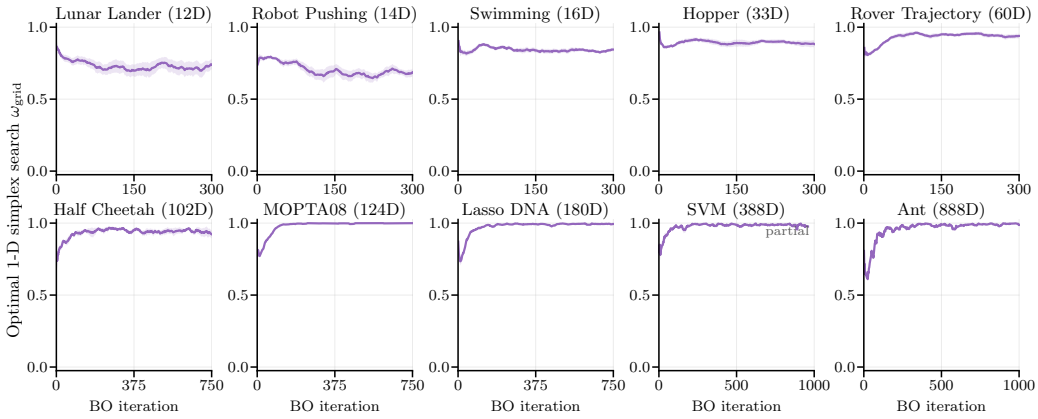


Figure 17: Optimal 1-D simplex search weight ω_{grid} for BOING-L with LogEI.

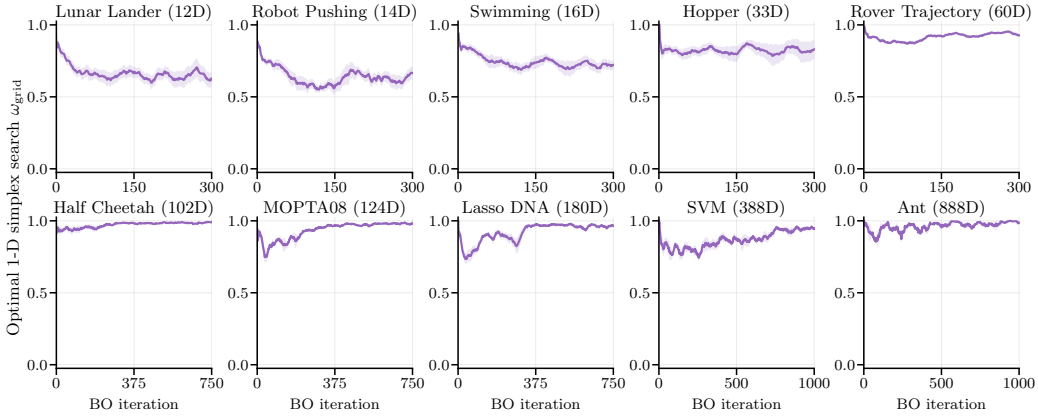


Figure 18: Optimal 1-D simplex search weight ω_{grid} for BOING-H with LogEI.

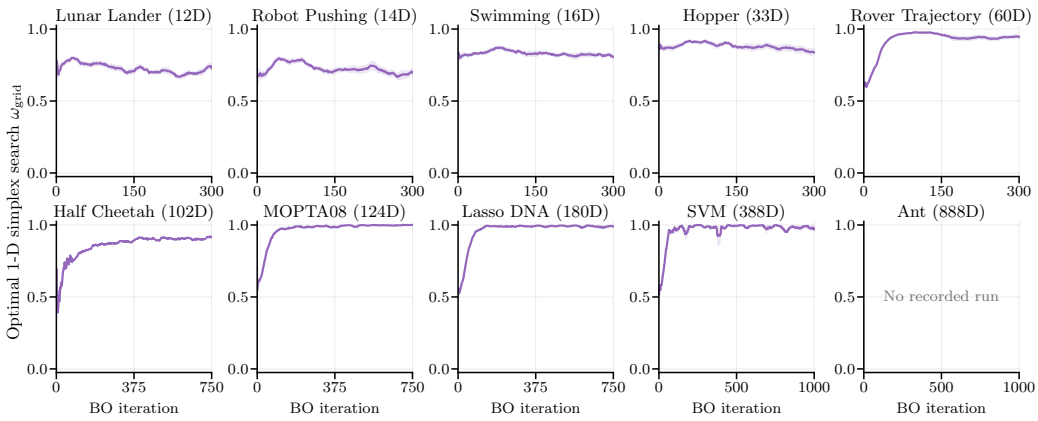


Figure 19: Optimal 1-D simplex search weight ω_{grid} for BOING-L with GIBBON.

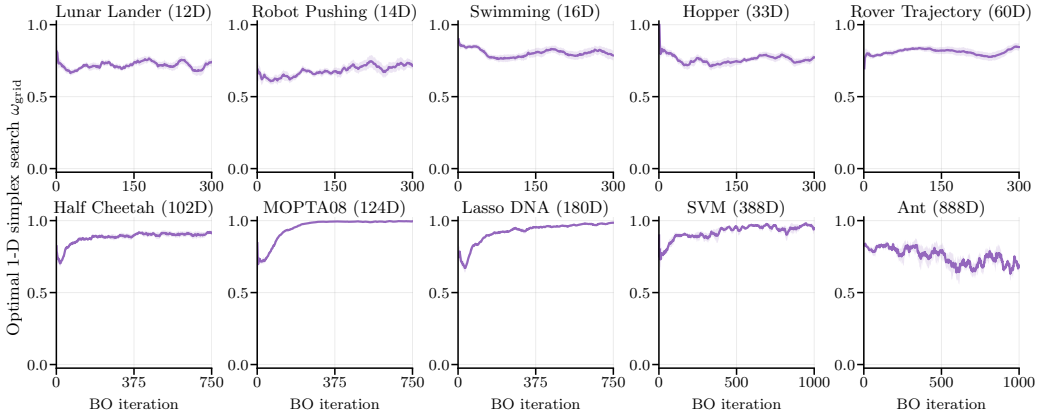


Figure 20: Optimal 1-D simplex search weight ω_{grid} for BOING-H with GIBBON.

C.7 Optimal Radius Multiplier

The optimal radius multiplier c in $\kappa_t(c) = c\overline{\lambda}_t$ is shown in Figures 21 and 22. In the recorded BOING-H LogEI curve choices, $c < 1$ is selected about 53% of the time on RobotPushing and 62% on HalfCheetah. The largest radius is therefore not always selected during BO, consistent with Figure 3. These choices show how the radius sweep selects different step scales during BO.

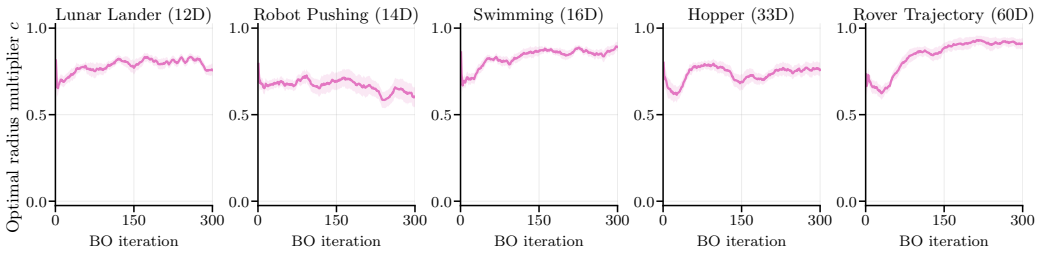


Figure 21: Optimal radius multiplier c in $\kappa_t(c) = c\overline{\lambda}_t$ for BOING-L with LogEI, for the configuration without median scaling.

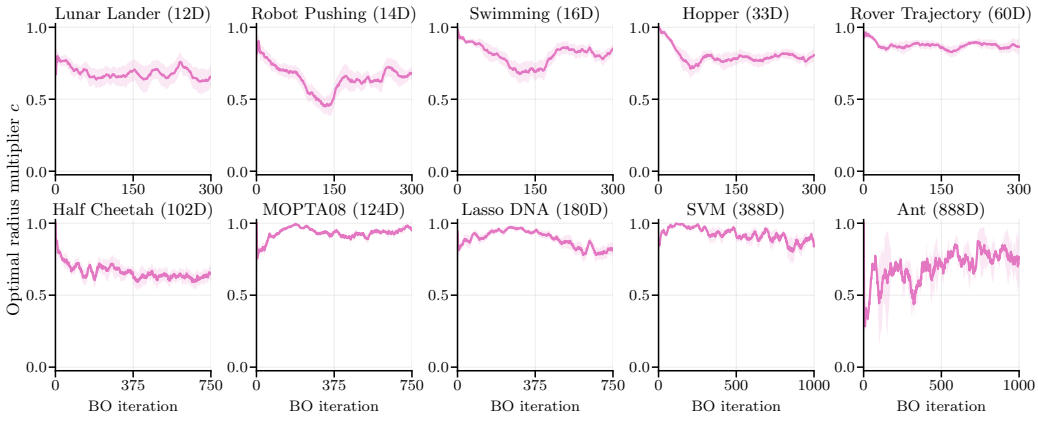


Figure 22: Optimal radius multiplier c in $\kappa_t(c) = c\overline{\lambda}_t$ for BOING-H with LogEI.

D Geometric Background and Optimization Derivations

D.1 Gaussian Predictive Distribution

The scalar Gaussian specialization used by BOING is recalled from Amari (2016).

Corollary D.1 (Pullback FIM for a Gaussian Predictive Distribution)**.** *Let Assumption 4.1 hold and suppose the predictive distribution is univariate Gaussian, $p(f(\mathbf{x})|\mathcal{D}_t) = \mathcal{N}\big(\mu_{t,\mathbf{x}}, \Sigma_{t,\mathbf{x}}\big)$, where $\mu_{t,\mathbf{x}} \in \mathbb{R}$ is the posterior mean and $\Sigma_{t,\mathbf{x}} > 0$ is the posterior variance, parameterized by $\boldsymbol{\theta}_{t,\mathbf{x}} = \big(\mu_{t,\mathbf{x}}, \Sigma_{t,\mathbf{x}}\big) \in \mathcal{S} = \mathbb{R} \times \mathbb{R}_+$. The Fisher–Rao metric at this point on the predictive distribution manifold is*

$$\mathcal{I}\big(\boldsymbol{\theta}_{t,\mathbf{x}}\big) = \operatorname{diag}\big(\Sigma_{t,\mathbf{x}}^{-1},\ \tfrac{1}{2}\,\Sigma_{t,\mathbf{x}}^{-2}\big)\,, \tag{24}$$

and the pullback tensor (2) *reduces to*

$$\mathbf{g}_t(\mathbf{x}) = \frac{\nabla\mu_{t,\mathbf{x}}\,\nabla\mu_{t,\mathbf{x}}^\top}{\Sigma_{t,\mathbf{x}}} + \frac{\nabla\Sigma_{t,\mathbf{x}}\,\nabla\Sigma_{t,\mathbf{x}}^\top}{2\,\Sigma_{t,\mathbf{x}}^2}, \tag{25}$$

where ∇ denotes the gradient with respect to $\mathbf{x}$.

The first term, $\nabla\mu_{t,\mathbf{x}}\nabla\mu_{t,\mathbf{x}}^\top/\Sigma_{t,\mathbf{x}}$, is large where the posterior mean changes rapidly relative to uncertainty, so it captures the local exploitation geometry. The second term, $\nabla\Sigma_{t,\mathbf{x}}\nabla\Sigma_{t,\mathbf{x}}^\top/(2\Sigma_{t,\mathbf{x}}^2)$, is large where the posterior variance changes rapidly, so it captures the local exploration geometry.

Equation (24) follows from the Gaussian score moments. Substitution in (2) gives (25).

D.2 Maximum Lies on the Fisher Boundary

For a starting location and an admissible radius. The local problem in (5) can be written over input displacements as

$$\underset{\boldsymbol{d}\in\operatorname{range}(J_r^\top)}{\text{maximize}} \quad \bar{\alpha}(\boldsymbol{\theta}_r + J_r\boldsymbol{d})\,, \qquad \boldsymbol{d}^\top\mathbf{g}_r\boldsymbol{d} \le \kappa^2.$$

Proof. Suppose that $\boldsymbol{d}_r^\star(\kappa)$ lies in the relative interior of the Fisher region. Then there exists an ϵ-ball, with $\epsilon > 0$, around it in $\operatorname{range}(J_r^\top)$ that remains feasible. Assumption 4.2 gives a sufficiently small positive predictive increment in this neighbourhood and therefore a feasible displacement $\boldsymbol{d}'$ with

$$\bar{\alpha}(\boldsymbol{\theta}_r + J_r\boldsymbol{d}') > \bar{\alpha}(\boldsymbol{\theta}_r + J_r\boldsymbol{d}_r^\star(\kappa))\,.$$

This contradicts that $\boldsymbol{d}_r^\star(\kappa)$ is optimal. Hence the optima must lie on the boundary of the Fisher constraint as

$$\big(\boldsymbol{d}_r^\star(\kappa)\big)^\top\mathbf{g}_r\boldsymbol{d}_r^\star(\kappa) = \kappa^2.$$

□

D.3 KKT Characterization of the Nonlinear Fisher Problem

For a starting predictive parameter $\boldsymbol{\theta}_r$ and a radius satisfying Proposition 4.3. Assumption 4.2 keeps this Fisher ellipsoid inside $\mathcal{S}$. Assumption 4.3 makes $\operatorname{range}(J_r)$ the full predictive parameter space. The Lagrangian of the nonlinear local problem is

$$\mathcal{L}(\boldsymbol{\Delta\theta}, \eta) = \bar{\alpha}(\boldsymbol{\theta}_r + \boldsymbol{\Delta\theta}) - \eta\big(\boldsymbol{\Delta\theta}^\top\mathcal{I}_r\boldsymbol{\Delta\theta} - \kappa^2\big)\,. \tag{26}$$

Since $\mathcal{I}_r \succ 0$ and $\kappa > 0$, the gradient of the active constraint is nonzero. Every constrained maximizer therefore satisfies the KKT conditions

$$\nabla_{\boldsymbol{\theta}}\bar{\alpha}(\boldsymbol{\theta}_r^\star(\kappa)) - 2\eta_r^\star(\kappa)\mathcal{I}_r\boldsymbol{\Delta\theta}_r^\star(\kappa) = \mathbf{0}, \qquad \text{stationarity,} \tag{27}$$

$$\big(\boldsymbol{\Delta\theta}_r^\star(\kappa)\big)^\top\mathcal{I}_r\boldsymbol{\Delta\theta}_r^\star(\kappa) - \kappa^2 \le 0, \qquad \text{primal feasibility,} \tag{28}$$

$$\eta_r^\star(\kappa) \ge 0, \qquad \text{dual feasibility,} \tag{29}$$

$$\eta_r^\star(\kappa)\Big[\big(\boldsymbol{\Delta\theta}_r^\star(\kappa)\big)^\top\mathcal{I}_r\boldsymbol{\Delta\theta}_r^\star(\kappa) - \kappa^2\Big] = 0, \qquad \text{complementary slackness.} \tag{30}$$

Proposition 4.3 makes the primal constraint active. Assumption 4.2 makes the predictive acquisition gradient nonzero. Stationarity then gives $\eta_r^\star(\kappa) > 0$ and

$$\boldsymbol{\Delta\theta}_r^\star(\kappa) = \frac{1}{2\eta_r^\star(\kappa)} \mathcal{I}_r^{-1} \nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r^\star(\kappa)) \,, \tag{31}$$

$$\eta_r^\star(\kappa) = \frac{1}{2\kappa} \sqrt{\left[\nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r^\star(\kappa))\right]^\top \mathcal{I}_r^{-1} \nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r^\star(\kappa))}. \tag{32}$$

Substitution gives

$$\boldsymbol{\Delta\theta}_r^\star(\kappa) = \frac{\kappa \mathcal{I}_r^{-1} \nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r^\star(\kappa))}{\sqrt{\left[\nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r^\star(\kappa))\right]^\top \mathcal{I}_r^{-1} \nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r^\star(\kappa))}}. \tag{33}$$

The gradient on the right is evaluated at the unknown optimum. This is an implicit necessary condition for the nonlinear problem, rather than a closed-form solution obtained from the starting gradient.

D.4 Linearized Fisher Alternative

For a starting predictive parameter $\boldsymbol{\theta}_r = \varphi_t(\mathbf{x}_r)$, Assumption 4.2 ensures that its predictive acquisition gradient is nonzero. The first-order approximation of the objective function in (5) is

$$\bar{\alpha}(\boldsymbol{\theta}_r + \boldsymbol{\Delta\theta}) \approx \bar{\alpha}(\boldsymbol{\theta}_r) + \left[\nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r)\right]^\top \boldsymbol{\Delta\theta}. \tag{34}$$

Assumption 4.3 makes the reachable predictive space full, so the range constraint does not change this linearized problem. The Lagrangian with the quadratic constraint is

$$\mathcal{L}_{\eta \geq 0}(\boldsymbol{\Delta\theta}, \eta) = \bar{\alpha}(\boldsymbol{\theta}_r) + \left[\nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r)\right]^\top \boldsymbol{\Delta\theta} - \eta\left(\boldsymbol{\Delta\theta}^\top \mathcal{I}_r \boldsymbol{\Delta\theta} - \kappa^2\right). \tag{35}$$

Differentiating (35) with respect to $\boldsymbol{\Delta\theta}$ and equating to $\mathbf{0}$ for optima,

$$\nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r) - 2\eta \mathcal{I}_r \boldsymbol{\Delta\theta} = \mathbf{0}, \tag{36}$$

For $\eta > 0$, the maximizing update of the Lagrangian is

$$\boldsymbol{\Delta\theta} = \frac{1}{2\eta} \mathcal{I}_r^{-1} \nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r) \,. \tag{37}$$

For $\eta > 0$, substituting (37) into (35) gives the dual problem

$$\min_{\eta > 0} \max_{\boldsymbol{\Delta\theta}} \mathcal{L}_{\eta \geq 0}(\boldsymbol{\Delta\theta}, \eta) = \min_{\eta > 0} \left\{ \bar{\alpha}(\boldsymbol{\theta}_r) + \eta\kappa^2 + \frac{1}{4\eta} \left[\nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r)\right]^\top \mathcal{I}_r^{-1} \nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r) \right\}. \tag{38}$$

Differentiating the scalar objective in (38) gives

$$0 = \kappa^2 - \frac{1}{4\eta^2} \left[\nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r)\right]^\top \mathcal{I}_r^{-1} \nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r) \,,$$

$$\eta^\star = \frac{1}{2\kappa} \sqrt{\left[\nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r)\right]^\top \mathcal{I}_r^{-1} \nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r)}. \tag{39}$$

Substituting $\eta^\star$ into (37) gives the optimal update for linearized problem as

$$\boldsymbol{\Delta\theta}_{\text{lin}}^\star(\kappa) = \frac{\kappa \mathcal{I}_r^{-1} \nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r)}{\sqrt{\left[\nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r)\right]^\top \mathcal{I}_r^{-1} \nabla_{\boldsymbol{\theta}} \bar{\alpha}(\boldsymbol{\theta}_r)}}. \tag{40}$$

Equation (40) is the normalized steepest-ascent direction for a quadratic norm (Boyd & Vandenberghe, 2004, Sec. 9.4.1). The same linearization and Lagrangian calculation is used by Fisher SAM (Kim et al., 2022, Eqs. 8–10). It uses the gradient at the starting predictive parameter and is not used by the proposed simplex sweep.

D.5 Geometric Derivation of the Simplex Update

For a starting predictive state $\boldsymbol{\theta} = (\mu, \Sigma)^\top \in \mathcal{S}$, the FIM is $\mathcal{I}(\boldsymbol{\theta}) = \operatorname{diag}(\Sigma^{-1}, (2\Sigma^2)^{-1})$. For the ease of notation we avoid the indices (t, r). Equation (10) places every normalized acquisition gradient in the monotonic neighbourhood $\kappa < \kappa^{\text{loc}}$ on the 1-D simplex. At a optima, the KKT conditions (9) shows that the gradient is parallel to the normal of the constraint. Since the optima

in (9) is rather implicit and we are able to show that the normalized gradient at the optima lies somewhere on the 1-D simplex, a simple search over the $\omega \in [0, 1]$ and finding the maxima of $\bar{\alpha}(\boldsymbol{\theta} + \boldsymbol{\Delta\theta}(\kappa, \omega))$ as mentioned in (11) and (12) can give us the optimal $\boldsymbol{\Delta\theta}^\star$.

The update $\boldsymbol{\Delta\theta}(\kappa, \omega)$ from the center $\boldsymbol{\theta}$ such that $\boldsymbol{\Delta\theta}(\kappa, \omega)$ is on the ellipse $\boldsymbol{\Delta\theta}^\top \mathcal{I}(\boldsymbol{\theta})\boldsymbol{\Delta\theta} - \kappa^2 = 0$ can be calculated by finding the normal direction for each $\omega \in [0, 1]$ and find the corresponding boundary point. The constraint normal is $\nabla_{\boldsymbol{\Delta\theta}} \left(\boldsymbol{\Delta\theta}^\top \mathcal{I}(\boldsymbol{\theta})\boldsymbol{\Delta\theta} - \kappa^2\right) = 2\mathcal{I}(\boldsymbol{\theta})\boldsymbol{\Delta\theta}$. Requiring it to point along the chosen direction gives

$$\begin{aligned} \mathcal{I}(\boldsymbol{\theta})\boldsymbol{\Delta\theta} &= c_{\text{normal}} \begin{bmatrix} \omega \\ 1-\omega \end{bmatrix}, \qquad c_{\text{normal}} > 0, \\ \boldsymbol{\Delta\theta} &= c_{\text{normal}} \mathcal{I}(\boldsymbol{\theta})^{-1} \begin{bmatrix} \omega \\ 1-\omega \end{bmatrix}. \end{aligned} \tag{41}$$

Here c_{normal} is a some positive proportaionality constant for the given Fisher radius. Substituting (41) into the constraint equation yields

$$\kappa^2 = c_{\text{normal}}^2 \begin{bmatrix} \omega & 1-\omega \end{bmatrix} \mathcal{I}(\boldsymbol{\theta})^{-1} \begin{bmatrix} \omega \\ 1-\omega \end{bmatrix}, \tag{42}$$

$$\implies c_{\text{normal}} = \frac{\kappa}{\sqrt{\begin{bmatrix} \omega & 1-\omega \end{bmatrix} \mathcal{I}(\boldsymbol{\theta})^{-1} \begin{bmatrix} \omega \\ 1-\omega \end{bmatrix}}}. \tag{43}$$

Substituting (42) into (41) gives

$$\boldsymbol{\Delta\theta}(\kappa, \omega) = \frac{\kappa \mathcal{I}^{-1} \begin{bmatrix} \omega \\ 1-\omega \end{bmatrix}}{\sqrt{\begin{bmatrix} \omega & 1-\omega \end{bmatrix} \mathcal{I}^{-1} \begin{bmatrix} \omega \\ 1-\omega \end{bmatrix}}}. \tag{44}$$

Assumption 4.1 ensures that the Fisher matrix is positive definite, hence the denominator is strictly positive for every $\omega \in [0, 1]$. Equation 44 is the unique boundary point for the constraint normal. Equivalently it can also be explained using support function where (44) is the point maximizing $[\omega, 1-\omega]\boldsymbol{\Delta\theta}$ with the Fisher ellipse as the boundary (Boyd & Vandenberghe, 2004, Example 3.7). The AF $\bar{\alpha}$ is then evaluated at these points and the maxima from (11) and (12) is the optimizer for $\bar{\alpha}$ as well; without linearizing the AF within the ellipse contraint.