Bayesian Optimization on Function Spaces via Sparse RKHS Manifolds

Davide Sartor
Department of Mathematics and Statistics
University of Massachusetts Amherst
dsartor@umass.edu

Meghan E. Huber
Department of Mechanical
and Industrial Engineering
University of Massachusetts Amherst
mehuber@umass.edu

Donghyun Kim
Manning College of Information
and Computer Sciences
University of Massachusetts Amherst
donghyunkim@cs.umass.edu

Nathan Wycoff
Department of Mathematics and Statistics
University of Massachusetts Amherst
nwycoff@umass.edu

Abstract

Bayesian Optimization (BO) has become an established methodology for minimizing black-box functions of a vector input. Often, however, this parameter vector arises from the discretization of an inherently functional relationship. Several recent articles have considered the Functional Bayesian Optimization (FBO) setting, in which the variable to be optimized is not a member of a finite dimensional vector space, but rather an infinite dimensional function space. In this work, we propose L^0 Manifold Optimization (L0MO), a simple approach to FBO which searches the subset of a Reproducing Kernel Hilbert Space (RKHS) consisting of functions with a sparse representation in the kernel functions, optimizing both the kernel locations and their coefficients. We discuss in detail the relationship between our method and existing ones, providing a unifying lens through which to view prior works. To assess our method against the state of the art, we conduct an extensive computational study, and along the way develop a novel set of benchmark test functions which port standard finite-dimensional ones to the infinite dimensional domain. Our experiments demonstrate that, on balance, the proposed method achieves superior performance across a wide range of test benchmarks.

1 Introduction

Many critical applied tasks across chemistry [35, 42], materials science [13, 30, 38] and biomechanics [22], among other areas, can be formulated as *black-box optimization* problems of the form:

$$\min_{u \in \mathcal{U}} f(u)\,, \tag{1}$$

where the domain $\mathcal{U}$ of f is usually a compact subset of $\mathbb{R}^P$, most commonly the hypercube $[0,1]^P$. A leading strategy for addressing these problems is BO [19, 12, 14], which involves reasoning probabilistically about the location of the global minimum given a set of observed input-output relationships $(u_1, f(u_1)), \ldots, (u_N, f(u_N))$. Given a small initial sample and a total budget B, the goal is to choose points one-by-one so as to find the point u with the lowest possible $f(u)$.

The finite dimensional setting is appropriate when f depends on a finite set of clearly defined parameters. Frequently, however, the parameters are actually a discretization of an intrinsically

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

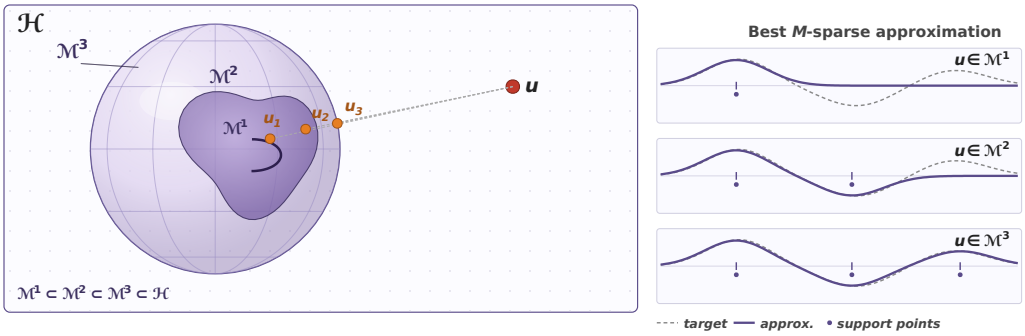

Figure 1: A schematic view inside $\mathcal{H}$, illustrating our approximation strategy. The sparse manifolds form a stratified set of increasing dimension, depicted as a curve, a surface patch, and a ball. For a target $u \in \mathcal{H}$ that does not admit a finite sparse representation, the best m-sparse approximation $u_m \in \bigcup_{i \leq m} \mathcal{M}^i$ moves closer to u as m grows.

functional quantity. For example, the classical *wing weight function* [11, Ch. 1] is typically viewed as an 11 dimensional parametric function. But these parameters encode the shape of a wing, and the space of possible designs actually consists of the in-principle infinite dimensional space of wing shapes.

Many other scientists have effected similar discretizations: subject area expertise is used to tame the infinite dimensional nature of the problem and therefore make it amenable to the available BO methodology and software environment. However, in so doing, they have also restricted the solution space before the campaign has begun: not only is it possible that some slight improvement over the given discretization could be obtained if a flexible functional form were allowed, this conversion to a finite-dimensional setting means that we restrict BO to learning solutions already conceivable by the subject matter expert. But this rules out the most exciting possibility: that there is some function of a radically different nature than the state of the art which yields unimagined decrease in f.

This explains the nascent interest in the research community in FBO. There, the search space $\mathcal{U}$ is a subset of some space $\mathcal{H}$ of $\mathcal{D} \to \mathbb{R}$ functions. For the sake of clarity we will use the term *function* to refer to the input $u \in \mathcal{U}$, and the term *functional* to refer to f. We are aware of only a few studies which directly consider FBO, and there remains much work to be done in studying this category of problems. [36] and [24] search within random linear subspaces, yielding a finite dimensional problem in effect. [39] search the space of n-th degree Bernstein polynomials, allowing the enforcement of shape priors. They propose a strategy for choosing the polynomial degree, but it is not clear how to scale the approach beyond one-dimensional functions. [40] show that starting with some initial design of functions and then iteratively conducting Bayesian acquisition is tractable, insofar as the trajectory obtained by gradient descent in the function space stays within the span of the kernel functions defining the initial design. These methods all assume that a good function can be found within the initially specified linear subspace.

In this article, we will propose to solve the acquisition subproblem of the form:

$$\max_{w \in \mathcal{W} \subseteq \mathbb{R}^l} \mathcal{I}\left(\phi(w)\right) , \tag{2}$$

where $\mathcal{I}$ is the acquisition function and $\phi : \mathbb{R}^l \to \mathcal{H}$ is a smooth parameterization of a stratified set in $\mathcal{H}$, obtained as the union of manifolds $\mathcal{M}$ of dimension at most l. For a given l, this allows a much richer set of candidate functions than existing fixed-basis method: our approach allows the support points themselves to vary. This also allows for straightforward enforcement of additional desirable properties such as boundedness and sparsity, while maintaining compatibility with the computational tools used for finite-dimensional BO such as multi-start Quasi-Newton search [15, Chapter 7].

Our contributions can be summarized as follows:

1. We provide an extensive discussion of disparate existing methods for FBO, providing a unifying lens and showing that [40] can be interpreted as natural gradient descent in a finite-dimensional parameter space.

2. We introduce a ridge function [31] approach to defining FBO problems from finite dimensional BO problems, which allows the use of test problems familiar to the BO community. We release an open source python implementation of $48+$ popular BO benchmarks and their extension to the functional domain, as well as several new realistic FBO problems.
3. We introduce a new method for FBO, which we call L0MO. We conduct an extensive simulation campaign across FBO hyperparameters and a range of synthetic and real test problems, finding that the proposed method performs best overall. We also release the first open source codebase for FBO at `https://github.com/davidesartor/bofus`.

2 Background

Gaussian Processes and Bayesian Optimization Gaussian processes [33] are a rule for assigning to every finite subset of points $\mathcal{X}$ in a domain $\mathcal{D}$ a joint normal distribution $\mathcal{N}(\mu(\mathcal{X}), \Sigma(\mathcal{X}))$. The mean function μ is typically taken to be a simple polynomial or even a constant (we do the latter in our numerical experiments), and determining Σ is the subject of most modeling decisions. Each element $\Sigma_{i,j}$ is given by a *kernel function* $\kappa(x, x')$; the set of functions yielding valid covariance matrices are the *positive definite functions*. κ is often given as a monotonically decreasing function of some measure of dissimilarity between its two arguments. In the most common application of GPs to vector inputs, the Radial Basis Function (RBF) kernel $\kappa_{\text{RBF}}(x, x') = \sigma^2 e^{-\frac{1}{2\ell}\|x-x'\|_2^2}$ is used. Note the presence of the hyperparameters σ^2 and ℓ, which must be learnt, most commonly via numerical likelihood maximization, and this is also the approach taken in this article. A key aspect of Gaussian processes that makes them practical is the tractability of their conditional distributions. If we observe the value of a GP realization at some set of points $\mathcal{X}_{\text{Train}}$, then the posterior distribution is again a Gaussian process with updated mean and covariance functions, straightforwardly computable via the multivariate normal conditioning equations.

The capability to produce a full posterior over any new candidate point x^* given an observation set $\mathcal{X}, f(\mathcal{X})$ undergirds the suitability of a Gaussian Process (GP) to perform sequential design. In BO, this proceeds by defining an *acquisition function* $\mathcal{I}$ which measures the desirability of x^* in light of the goal of minimization and the information revealed by the data so far observed. A popular acquisition function, and the one we use in this work is Expected Improvement (EI):

$$\mathcal{I}_{\text{EI}}(u \mid \mathcal{GP}) = \sigma_{\mathcal{GP}}(u)\big(\gamma(u)\Phi(\gamma(u)) + \varphi(\gamma(u))\big), \quad \gamma(u) = \frac{y^* - \mu_{\mathcal{GP}}(u)}{\sigma_{\mathcal{GP}}(u)}, \tag{3}$$

where y^* is the current best observation and Φ and φ are the standard normal CDF and PDF. The next point to acquire is the one which maximizes the acquisition function. Numerically resolving this is no small task, and typically involves a multi-start quasi-Newton search.

Gaussian Processes on Function Spaces Given a kernel function k, its RKHS is the closure of all functions of the form $u = \sum_{i=1}^n \alpha_i k(\cdot, x_i)$ for any n, equipped with an inner product which privileges these kernel functions. In other words, given a *support set* of functions $\mathcal{F} := \{k(\cdot, x) : x \in \mathcal{D}\}$, the RKHS $\mathcal{H}$ is $\overline{\text{span}(\mathcal{F})}$. For any two functions expressible as a finite linear combination of support functions, $u = \sum_{i=1}^n \alpha_i k(\cdot, x_i)$ and $u' = \sum_{j=1}^m \alpha'_j k(\cdot, x'_j)$, we have that $\langle u, u' \rangle_{\mathcal{H}} = \sum_{i,j} \alpha_i \alpha'_j k(x_i, x'_j)$, and extend $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ to the whole space by continuity.

As GPs rely on the domain on which they are fit only via κ, they are applicable to a wide range of mathematical objects, including functions. In fact, most kernels can be decomposed into a dissimilarity measure, composed with a *profile* function. In particular, given some Hilbert space of functions $\mathcal{H}$ with norm $\|\cdot\|_{\mathcal{H}}$, we can populate the covariance matrix entry between two functions $u_i, u_j \in \mathcal{H}$ via $\Sigma_{i,j} = p(\|u_i - u_j\|_{\mathcal{H}})$ for a given profile function p. In general, this requires approximating the norm on the function space [27, 3, 8]. Following [40], we use an RKHS as our function space. Since we are in the context of a designed experiment rather than observational data, we can afford to choose functions which lead to an easily computed $\|\cdot\|_{\mathcal{H}}$. In particular, if two functions $u, u' \in \mathcal{H}$ are expressible with a finite linear combination of elements of $\mathcal{F}$ we have that

$$\|u - u'\|_{\mathcal{H}}^2 = \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j) - 2 \sum_{i,j} \alpha_i \alpha'_j k(x_i, x'_j) + \sum_{i,j} \alpha'_i \alpha'_j k(x'_i, x'_j), \tag{4}$$

which can be computed efficiently if neither function consists of overly many basis points.

RKHS's and Gaussian processes are intimately linked [20]: for example, the posterior mean of a Gaussian process regression lives in the RKHS induced by the kernel used for the GP. It will be important to avoid confusion between $\kappa : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$, the kernel of the GP used as surrogate model, which accepts two functions in $\mathcal{H}$ as input and outputs an element of the covariance of the Gaussian process, and $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, the kernel of the RKHS that represents the functional domain $\mathcal{D}$ of interest, and which is used to construct our candidate functions u.

In this work, it will be useful to define an extension of the ℓ^0 "norm" [1] for vectors, which counts its argument's nonzero coefficients, to functions in an RKHS generated by $\mathcal{F}$:

$$\|u\|_0 = \inf \left\{ m \in \mathbb{N} : \exists\, \alpha_1, \ldots, \alpha_m \in \mathbb{R},\ x_1, \ldots, x_m \in \mathcal{D} \text{ such that } u = \textstyle\sum_{i=1}^m \alpha_i k(\cdot, x_i) \right\}, \quad (5)$$

with $\inf \emptyset = \infty$. We refer to this as the L^0 norm. The functions of finite L^0 norm form the pre-RKHS $\mathcal{H}_0 = \text{span}(\mathcal{F})$, which is dense in $\mathcal{H}$ by construction. Notice that while the vector constraint set $\{w \in \mathbb{R}^P : \|w\|_0 \leq m\}$ is a union of coordinate subspaces that meet only at low-sparsity points, in $\mathcal{H}$ the support locations x_i can move continuously, so that any two functions with $\|u\|_0 \leq m$ are connected by a path within the set; this facilitates optimization.

3 Related Work in FBO

BO is classically deployed in low dimensional settings ($P \approx 20$). However, recently methods have been proposed which aim to make BO more amenable to higher dimensional problems. These include using random embeddings [43, 26, 28], additive structure [21, 34, 16], and variable selection [29, 10]. Recent works have also re-examined the amenability of simple BO strategies to scale to high dimensions [18, 9]. In this section, we rather review the established literature on the $P = \infty$ case. As mentioned in the introduction, the overwhelming majority of optimization studies using BO to optimize over functions first assume a finite dimensional parameterization before proceeding with standard methods. Here we provide an overview of studies which have instead attempted to treat the function space directly, and which will serve as baselines in the numerical experiments.

Explicit Subspace Methods [24] compare two methods which operate by searching a $K < \infty$ dimensional linear subspace $\mathcal{S} \subset \mathcal{H}$ generated by the span of given functions $s_1, \ldots, s_K$. The first generates these functions via a random sampling mechanism on $\mathcal{H}$, and the second uses the leading eigenfunctions of an integral operator defined with reference to the objective f. However, the latter procedure assumes that functional gradients of f can be evaluated, which is not available in our black-box setting. We therefore focus on exploring the span of random functions.

[36] propose a more sophisticated method which searches within a sequence of affine subspaces. Given a current optimal function u_0, it generates a random set of functions $s_1, \ldots, s_K$ and searches within the K dimensional space $u_0 + \text{span}(\{s_1, \ldots, s_K\})$ for a fraction ρB of the overall budget B. Then, it resamples a new set $s_1, \ldots, s_K$ and searches within the span of those new points offset by the new optimal u. This avoids choosing a single finite dimensional subspace to search within, and operates somewhat like an infinite dimensional analog of [23]. [39] take a parametric approach, using Bernstein polynomials of degree k as candidate functions. The Bernstein basis allows for the easy enforcement of shape priors, such as monotonicity. However, the Bernstein parameterization is most natural in one dimension, and it is unclear how to extend the approach to higher dimensional $\mathcal{D}$.

Direct Optimization in the Function Space [40] showed that, when EI or UCB is maximized by gradient descent in $\mathcal{H}$ starting from a candidate u_0 with finite L^0 norm, every iterate remains in the span of the support functions of u_0 and of the N previously observed functions $u_1, \ldots, u_N$ (eq. (16)), so that only the coefficients of those support functions evolve. This is a positive result in a computational sense: it could have been the case that a single step of gradient descent yields a function which does not have a finite L^0 norm, and thus cannot be represented on a computer. Luckily, however, we need only update coefficients.

However, this is a double-edged sword: the support locations are never optimized, and are instead fixed by the random initialization u_0 of each acquisition search and by the past observations. Expressivity must therefore come from the number of support functions: the random initializations must have a high

[1] As it clearly fails to obey positive homogeneity, this is not technically a norm. The same will be true of our extension.

L^0 norm, denoted v, and every candidate additionally inherits the supports of all past observations. As the number of observations N grows, the complexity of computing the GP kernel matrix grows with v^2N^2. [40] find that, to manage computation, it is therefore necessary to conduct a post-hoc sparsification step after optimizing the acquisition function. This involved deploying Kernel Matching Pursuit (KMP) [41] to reduce to only $m \ll v$ many support functions. Then, kernel computation is only of complexity $v^2 N_{\text{init}}^2 + vmN_{\text{init}}N + m^2N^2 = O(m^2N^2)$, where N_{init} is the size of the initial design.

There are two major limitations to this approach. First, even if the initial points have a high L^0 norm, there is no guarantee that they are suitable for approximately optimizing f, especially if $\mathcal{D}$ is of moderate dimension. Though motivated as performing optimization directly in $\mathcal{H}$, each acquisition of [40] is ultimately restricted to a finite dimensional linear subspace of $\mathcal{H}$, spanned by support functions that are drawn at random or inherited from past observations, rather than chosen to improve the acquisition. Second, the post-hoc sparsification step creates a critical issue: the sparse approximation of the optimum is not necessarily the optimum among sparse candidates. This situation is similar to solving an integer program by first solving a continuous relaxation and then rounding. In the next section, we will propose a method which addresses both of these limitations.

4 Methodology

We propose to directly optimize over the set of functions we can feasibly represent, i.e. the ones with a sparse representation in $\mathcal{F}$, say of size m. However, we maintain and even improve expressibility by allowing the support functions themselves to be selected, not only their coefficients. This means jointly choosing $\alpha_1, \ldots, \alpha_m \in \mathcal{A}$ for a suitable coefficient region $\mathcal{A}$, as previous methods have, but also $x_1, \ldots, x_m \in \mathcal{D}$ to define a candidate function $u = \sum_{i=1}^m \alpha_i k(\cdot, x_i)$. For a given m, we therefore search the set $\mathcal{M}^m = \{u \in \mathcal{H} : \|u\|_0 \leq m\}$. This set is not a linear subspace: for instance, $k(\cdot, x_1)$ and $k(\cdot, x_2)$ both belong to $\mathcal{M}^1$, but $k(\cdot, x_1) + k(\cdot, x_2)$ does not in general. Nor is it a single manifold, but rather a stratified set $\mathcal{M}^m = \bigcup_{j \leq m} S^j$, whose strata $S^j = \{u \in \mathcal{H} : \|u\|_0 = j\}$ collect the functions of exact sparsity j. For $\mathcal{D} \subseteq \mathbb{R}^d$ and the RBF kernel, the canonical parameterization $\phi(\alpha, x) = \sum_{i=1}^m \alpha_i k(\cdot, x_i)$ maps the open set $W^m = \{(\alpha, x) : x_i \text{ pairwise distinct}, \ \alpha_i \neq 0\} \subset \mathbb{R}^{(d+1)m}$ onto the top stratum S^m as an $m!$-sheeted covering, the sheets differing only by a permutation of the support points. The top stratum is therefore an immersed smooth manifold of dimension $l := (d+1)m$, while the lower strata S^j, $j < m$, are reached at the boundary of W^m, where a coefficient vanishes or two support points coincide. The sets are nested such that $\mathcal{M}^1 \subset \mathcal{M}^2 \subset \ldots$, and their union is the pre-RKHS $\mathcal{H}_0$. For general $\mathcal{D}$, there are functions in $\mathcal{H}$ that do not belong to any $\mathcal{M}^m$. However, since $\mathcal{H}_0$ is dense in $\mathcal{H}$, any function in $\mathcal{H}$ can be approximated arbitrarily well by a member of some $\mathcal{M}^m$, and the optimal value of a continuous objective over $\mathcal{M}^m$ converges to its infimum over $\mathcal{H}$ as m grows. For moderate m, we therefore expect this method to perform well when the functional f can be approximately minimized using functions with small L^0 norm. From the function space perspective, at each iteration we solve:

$$\max_{u \in \mathcal{H}} \quad \mathcal{I}(u) \quad s.t. \quad \|u\|_0 \leq m\,, \tag{6}$$

which practically speaking is implemented as:

$$\max_{\substack{x_1,\ldots,x_m \in \mathcal{D} \\ \alpha_1,\ldots,\alpha_m \in \mathcal{A}}} \quad \mathcal{I}\left(\sum_{i=1}^m \alpha_i k(\cdot, x_i)\right)\,. \tag{7}$$

This involves solving a finite dimensional search to which we can apply the standard tools used to solve the BO acquisition subproblem. Over all of $\mathcal{H}$ the maximum need not exist, since sparse sets are not compact and the acquisition need not be continuous in their topology. In eq. (7), however, the search is over a compact parameter set (a compact $\mathcal{D}$ and a compact $\mathcal{A}$, e.g. the box constraints of the bounded parameterization below) through the continuous map ϕ, and the GP posterior mean and standard deviation, and hence EI, are continuous in the RKHS distance for the kernels we use, so that a maximizer exists by the extreme value theorem.

Natural Gradient on the Manifold. A subtlety arises when one optimizes over $\mathcal{M}^m$ via a parameterization $\phi : \mathbb{R}^l \to \mathcal{H}$. The geometry that matters for the acquisition functional lives in $\mathcal{H}$, but a naive optimizer in the parameter space $\mathbb{R}^l$ uses the Euclidean metric, which is in general unrelated

to $\mathcal{H}$-distance. Equal Euclidean steps in $\mathbb{R}^l$ correspond to unequal $\mathcal{H}$-displacements, so a standard Euclidean gradient takes a step that is not aligned with the steepest direction on the manifold. The classical remedy is the natural gradient [1]. Recall that gradient descent can be derived as the repeated solution of quadratic problems of the form:

$$w_{t+1} = \operatorname*{argmin}_w f(w_t) + \nabla f(w_t)^\intercal (w - w_t) + \frac{1}{2\lambda}\|w - w_t\|_2^2, \tag{8}$$

which yields the update $w_{t+1} = w_t - \lambda \nabla f(w_t)$. Replacing the squared Euclidean distance with a non-Euclidean dissimilarity $D(w, w_t)^2$ produces the natural gradient update

$$w_{t+1} = w_t - \lambda G^+ \nabla f(w_t), \tag{9}$$

where G is the metric tensor associated with D and G^+ its Moore–Penrose pseudoinverse.[2]

For a smooth parameterization $\phi : \mathbb{R}^l \to \mathcal{H}$, the natural choice of D is the $\mathcal{H}$-distance pulled back through ϕ: $D(w, w') = \|\phi(w) - \phi(w')\|_{\mathcal{H}}$. The resulting metric is the *pullback metric* with metric tensor

$$G_{ij}(w) = \langle \partial_i \phi_w, \partial_j \phi_w \rangle_{\mathcal{H}}, \tag{10}$$

which can be expressed as the cross-partial derivative of the RKHS inner product:

$$G_{ij}(w) = \frac{\partial}{\partial v_i} \frac{\partial}{\partial v'_j} \langle \phi(v), \phi(v') \rangle_{\mathcal{H}} \Big|_{v=v'=w}. \tag{11}$$

This formula reduces computation of the metric tensor to differentiating a scalar function of v and v', which can be handled by any autodifferentiation framework, regardless of the choice of parameterization.

Canonical Parameterization. The simplest concrete choice is the canonical representation with a fixed number m of support points:

$$\phi(w) = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i), \qquad w = (\alpha, x) \in \mathbb{R}^m \times \mathbb{R}^{m \times d}. \tag{12}$$

The RKHS inner product between two such functions is

$$\langle \phi(v), \phi(v') \rangle_{\mathcal{H}} = \alpha^\intercal K(x, x') \alpha', \tag{13}$$

where $K(x, x')_{ij} = k(x_i, x'_j)$. The pullback metric tensor has block structure $G = \begin{pmatrix} G_{\alpha\alpha} & G_{\alpha x} \\ G_{x\alpha} & G_{xx} \end{pmatrix}$, where each block is obtained by differentiating $\alpha^\intercal K(x, x')\alpha'$ in (α, x) and (α', x') .

Crucially, this parameterization keeps the candidate $\phi(w)$ sparse with exactly m support points at all times, and allows joint optimization over both coefficients and locations.

Bounded Parameterization. If additional properties are desired, such as boundedness of the candidate function, an alternative parameterization can be used. Instead of parameterizing directly by (α, x), we use an implicit parameterization (y, x) where $y \in [y_{\min}, y_{\max}]^m$ are the (approximate) function values at the support points. The coefficients are derived via

$$\alpha = (K_{xx} + \varepsilon I)^{-1} y. \tag{14}$$

This gives $\phi_w(x) = K_{xx}\alpha = K_{xx}(K_{xx} + \varepsilon I)^{-1} y \approx y$ when the support points are sufficiently far apart, while the ε-regularization shifts all eigenvalues of K_{xx} by ε to ensure that α is well defined even when K_{xx} is near-singular. Boundedness of $\phi(w)$ at the support points is enforced by constraining $y \in [y_{\min}, y_{\max}]^m$. The pullback metric for this parameterization follows from eq. (11).

Quasi-Newton Optimization. Although natural gradient descent provides a geometrically motivated update, the standard practice in BO is to use a quasi-Newton method such as L-BFGS-B [7], which builds an empirical preconditioner on top of the one provided by the natural gradient. We found this to give strong empirical performance on the resulting finite-dimensional problem (see section B). The bounded parameterization enforces box constraints on y, and modern autodifferentiation frameworks compute the gradient with respect to (y, x) at negligible cost. We use multi-start L-BFGS-B in all experiments.

[2] On the top stratum W^m the metric tensor of the canonical parameterization is positive definite and $G^+ = G^{-1}$. It degenerates at the boundary, where the parameterization stops being an immersion: if $\alpha_i = 0$, for example, any displacement of x_i leaves ϕ unchanged. The pseudoinverse discards these directions (see section B for a truncated variant).

Recovering the Updates of [40]. The proposed framework includes the gradient descent procedure of [40] as a special case. They show that for a surrogate GP with squared exponential kernel $\kappa(u, u') = e^{-\frac{1}{2\ell}\|u-u'\|_{\mathcal{H}}^2}$, the Fréchet derivative of κ with respect to u is the linear functional

$$D_u\kappa(u, u') = -\frac{\kappa(u, u')}{\ell}\langle \cdot, u - u'\rangle_{\mathcal{H}}, \tag{15}$$

whose Riesz representer in $\mathcal{H}$ is $-\frac{\kappa(u,u')}{\ell}(u - u')$. For $u = \sum_{i=1}^{m_u} \alpha_i k(\cdot, x_i)$ and $u' = \sum_{i=1}^{m_{u'}} \alpha'_i k(\cdot, x'_i)$, this representer lies in $\mathcal{H}$ with support points given by the union of those of u and u'. Since the gradient of the acquisition involves $\kappa(u, u_n)$ for every observed function u_n, $n = 1, \ldots, N$, after t steps of gradient descent from u_0 the candidate u_t accumulates the support points of all observed functions:

$$u_t = \sum_{i=1}^{m_{u_0}} \beta_i k(\cdot, x_i) + \sum_{n=1}^{N} \sum_{i=1}^{m_{u_n}} \beta_{n,i} k(\cdot, x_{n,i}), \tag{16}$$

where $x_{n,i}$ are the support points of u_n. To interpret this as natural gradient descent, fix the support set $\{x_i\}_{i=1}^m$ to the union of the supports of u_0 and of all observed functions, and consider the parameterization:

$$\phi(w) = \sum_{i=1}^{m} w_i k(\cdot, x_i), \qquad w \in \mathbb{R}^m. \tag{17}$$

The pullback metric tensor of eq. (10) for this parameterization is:

$$G_{ij} = \langle \partial_i\phi, \partial_j\phi\rangle_{\mathcal{H}} = \langle k(\cdot, x_i), k(\cdot, x_j)\rangle_{\mathcal{H}} = k(x_i, x_j), \tag{18}$$

i.e. G is the kernel matrix of k evaluated on the fixed support set. Substituting $u = \phi(w)$ and $u' = \phi(w')$, the squared norm reduces to $\|\phi(w) - \phi(w')\|_{\mathcal{H}}^2 = (w - w')^\top G(w - w')$, and the Euclidean gradient with respect to w is:

$$\nabla_w \kappa(\phi(w), \phi(w')) = -\frac{\kappa(\phi(w), \phi(w'))}{\ell}\, G(w - w'). \tag{19}$$

The update direction of [40], expressed as a coefficient vector in this parameterization, is instead $-\frac{\kappa(\phi(w),\phi(w'))}{\ell}(w - w')$, which is exactly the Euclidean gradient premultiplied by G^{-1} (or G^+ if support points repeat). The updates of [40] are therefore equivalent to natural gradient descent in this parameterization.

5 Experimental Results

We compare L0MO against VIEN [40], VELLANKI [39], KUNDU [24], and SHILTON [36], described in section 3, on function projection problems, on ridge functionals derived from classical BO benchmarks, and on real-world tasks including control problems and functional hyperparameter optimization. All objectives are minimized and their optimal value is not necessarily zero.

We investigate two metrics of interest: the best target value and average regret. For each metric, we report the median and 95% Confidence Interval (CI) across independent runs. All methods share the same budget and complexity schedule: after 10 initial designs, the size k of the acquisition problem grows from 1 to 10, increasing every 10 acquisitions, for a total of 110 evaluations (section F); section C compares this schedule with a fixed k. The optimal RKHS lengthscale and the GP kernel are selected for each method and benchmark (sections A and D). As a baseline, we report the value obtained by following the same schedule without optimizing the acquisitions (RANDOM), i.e. sampling from the RKHS at random. A breakdown of wall-clock times is given in section H.

Projections. As a simple toy problem, we project a fixed target function $\tilde{f}$ onto the RKHS, minimizing the squared L^2 distance between the candidate and the target. We use the *sinc* function, scaled and centered so that its main oscillations lie within the unit domain $[0, 1]^d$:

$$\tilde{f}(x) = \frac{1}{d}\sum_{i=1}^{d} \operatorname{sinc}(2\pi x_i - \pi)\,; \quad f(u) = \|\tilde{f} - u\|_{L^2}^2\,, \tag{20}$$

where the integral is computed with a midpoint quadrature on a tensor grid of about 4096 points. We consider input dimensions $d \in \{1, 2, 3, 4, 5\}$ to evaluate how the methods perform on a similar task at varying dimensionalities.

Ridge Functions. A large corpus of benchmark functions exists for standard BO, but comparatively few for FBO. We propose a systematic approach to adapt such benchmarks to the functional setting, extending the concept of *ridge functions* to the functional domain. Given a scalar benchmark $\bar{f} : [0,1]^d \to \mathbb{R}$ and a candidate function $u : [0,1]^d \to \mathbb{R}$, we define the functional benchmark as

$$f(u) = \bar{f}\big(\sigma\big(b + L[u]\big)\big), \qquad L_i[u] = \int_{[0,1]^d} w_i(x)\, u(x)\, dx, \quad i = 1, \dots, d, \tag{21}$$

where σ is the logistic function applied elementwise, $b \sim \mathcal{U}(-1,1)^d$ is a random bias, and each weight w_i is a random trigonometric polynomial with frequencies $\|\omega\|_1 \leq 4$ and decaying amplitudes, normalized to $\|w_i\|_{L^2} = 1$. The functionals L_i are not represented by a sparse element of $\mathcal{H}$, so we do not anticipate this construction to unduly favor our method. For RBF RKHS candidates the integrals factorize across dimensions and are computed in closed form via the complex error function; for other candidate representations we use a midpoint quadrature. Evaluating L_i by averaging $w_i u$ at n random points instead yields a Monte Carlo estimate with $O(1/\sqrt{n})$ error, and we verify this convergence in section G.

We provide a JAX [5] compatible open-source implementation of the virtual library of [37], comprising $48+$ benchmark functions, together with the functional ridge extension. In this work we consider five popular benchmarks of increasing input dimensionality: *GramacyLee* ($d = 1$), *Ackley* ($d = 2$), *Hartmann* ($d = 3$), *Rosenbrock* ($d = 4$), and *Michalewicz* ($d = 5$).

Real World Tasks. We next consider problems where the functional parameter has a direct physical or model interpretation. The *Brachistochrone* problem asks for the curve $y : [0,1] \to \mathbb{R}$ passing through two given points such that a bead placed at the start point and sliding along it under gravity reaches the end in the minimum time. The optimal solution is known to be a cycloid and the optimal travel time is known. Efforts to analytically solve the Brachistochrone problem precipitated the development of variational analysis, and we think this makes the Brachistochrone a suitable testbed for FBO. We optimize for the travel time in excess of that of the cycloid.

The second benchmark is the optimization of the activation function of a neural network. A better activation can speed up convergence and increase accuracy. We train a simple Multi Layer Perceptron (MLP) with two hidden layers of 256 neurons on the MNIST digit recognition task [25]. Our setup differs slightly from [40]: we measure the average test error across multiple training runs rather than a single seed, and instead of optimizing for the activation directly, we optimize an offset relative to the popular ReLU activation.

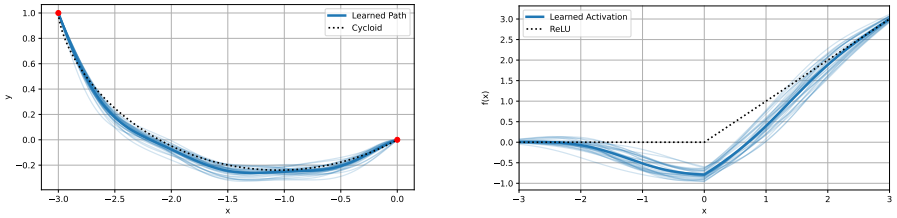

Figure 2: Best functions found by L0MO: brachistochrone paths (left) and MNIST activation functions, i.e. ReLU plus the learned offset (right). Transparent lines show individual runs; bold lines show the average across runs.

Robotics Control Tasks. We also consider three control tasks where the policy itself is the functional parameter being optimized. The first is the classic underactuated pendulum environment from OpenAI Gym [6], where the objective is to swing up and balance the pendulum by applying a torque $u : \mathbb{R}^2 \to \mathbb{R}$ as a function of the current angle and angular velocity. The objective is the expected cumulative reward over an episode, evaluated over 50 rollouts. This problem is more challenging than the typical Reinforcement Learning (RL) setting since we are optimizing the policy without access to state transitions; gradient-based policy optimization cannot be applied.

The second is a more complex environment in which a double-jointed robot arm must interact with a pinwheel and rotate it to a given target angle. The policy is a zero-force (i.e., desired) trajectory of the shoulder joint $u : [0,1] \to \mathbb{R}$ for a joint-space impedance controller [17]. The joint stiffness and damping parameters in the controller are fixed, as well as the zero-force equilibrium point of the

elbow joint. The objective is the cosine distance between the final pinwheel angle and the desired one. The arm must make and break contact with the pinwheel multiple times while respecting its kinematics; such hybrid problems, with both continuous motions and a discrete contact sequence, are challenging for standard solvers due to the discontinuity of the gradient at the contact transitions [32].

The third, *Hopper*, is a planar single-leg robot (body, thigh, shank and foot) with rigid ground contact and Coulomb friction, which must jump as high as possible. The policy is a zero-force (i.e., desired) trajectory of the hip and knee joints $u : [0, 1] \to \mathbb{R}^2$ over the 2 s episode for a joint-space impedance controller. The joint stiffness and damping parameters in the controller are fixed. The ankle was held at a fixed angle. The objective is the negative maximum height of the body after the first landing, in metres. As in the pinwheel task, intermittent contact makes the objective discontinuous in the policy.

Results. Figure 3 and table 1 report the results. On the projections, L0MO attains the lowest median average and final regret on Sinc-1D, Sinc-2D and Sinc-3D, and the lowest or tied median average regret on Sinc-4D and Sinc-5D, where VIEN [40] has a marginally lower final value. L0MO also attains the lowest median average and final regret on all five ridge functionals, with the largest margins on Ackley, Hartmann and Rosenbrock, where the competitors improve only moderately over random sampling. On the real-world tasks the gaps are smaller. L0MO has the lowest median on Pendulum, Pinwheel and Hopper, and also on MNIST, although there all methods lie within about a tenth of a percentage point of test error and the CIs of L0MO, KUNDU [24] and SHILTON [36] overlap. On Brachistochrone, SHILTON [36] reaches a slightly lower final regret, although L0MO has the lower median average regret. We attribute this to SHILTON [36]'s search being centered at the incumbent and spanned by smooth GP prior samples, which favors the local refinement of a smooth, non-sparse optimum such as the cycloid late in the run. Figure 2 shows the functions found by L0MO on Brachistochrone and MNIST.

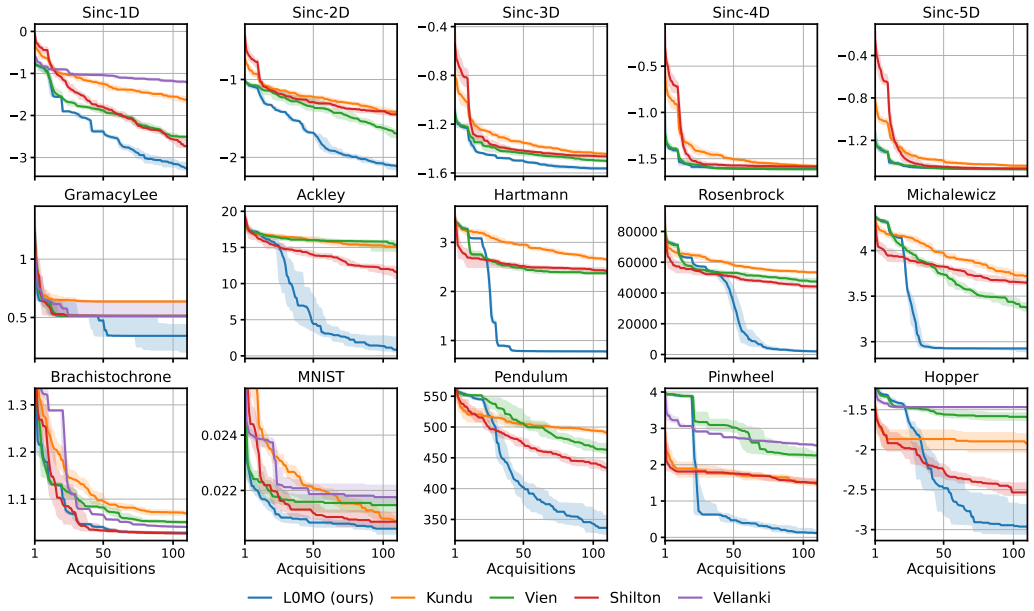

Figure 3: Method comparison across benchmark functions: projections (top), functional ridges (middle) and real-world tasks (bottom). Each plot shows the running best target function value (median and 95% CI) as a function of the number of acquisitions.

6 Discussion, Conclusion and Future Work

We proposed a new method for FBO which optimizes over a sparse subset of an RKHS, jointly searching over basis point locations and coefficients. We showed that the gradient descent procedure of [40] is equivalent to natural gradient descent in a finite-dimensional parameter space, and that this perspective generalizes to any smooth parameterization. We deployed our method on a range of synthetic and real-world benchmarks, where it attained the best or tied-best median performance on most of them.

Table 1: Average and final regret (median and 95% CI) across methods and benchmark functions. Columns group projections (left), functional ridges (middle) and real-world tasks (right). ✗ indicates the method does not support the corresponding problem class.

	Sinc-1D $[10^{-2}]$		**GramacyLee** $[10^{-1}]$		**Brachistochrone** $[10^{-2}]$	
Method	**Avg**	**Final**	**Avg**	**Final**	**Avg**	**Final**
Random	8.02 [6.26, 8.89]	6.67 [5.01, 7.43]	5.53 [5.38, 5.61]	5.19 [5.14, 5.24]	12.4 [11.8, 13.0]	9.80 [8.06, 11.5]
KUNDU [24]	8.48 [7.79, 8.69]	2.38 [2.06, 2.76]	6.56 [6.50, 6.65]	6.37 [6.37, 6.37]	10.8 [10.3, 11.3]	5.16 [4.86, 5.51]
VIEN [40]	2.58 [2.31, 2.82]	0.31 [0.26, 0.33]	5.45 [5.39, 5.53]	5.13 [5.13, 5.13]	7.42 [7.15, 7.83]	3.28 [3.11, 3.52]
SHILTON [36]	6.86 [6.16, 7.72]	0.19 [0.18, 0.26]	5.50 [5.38, 5.63]	5.13 [5.13, 5.13]	5.07 [3.91, 6.78]	**0.87** [**0.84**, **0.91**]
VELLANKI [39]	9.65 [9.62, 9.73]	6.24 [6.10, 6.46]	5.52 [5.41, 6.48]	5.13 [5.13, 6.37]	9.15 [8.98, 9.42]	2.26 [2.25, 2.29]
L0MO	**1.93** [**1.85**, **2.04**]	**0.06** [**0.05**, **0.07**]	**4.35** [**4.01**, **5.04**]	**3.42** [**2.08**, **4.35**]	**4.71** [**4.46**, **4.83**]	1.02 [1.01, 1.06]

	Sinc-2D $[10^{-2}]$		**Ackley** $[10^{1}]$		**MNIST** $[10^{-2}]$	
Method	**Avg**	**Final**	**Avg**	**Final**	**Avg**	**Final**
Random	6.80 [6.52, 7.20]	6.38 [6.01, 6.99]	1.58 [1.55, 1.59]	1.46 [1.43, 1.51]	2.18 [2.15, 2.19]	2.16 [2.14, 2.18]
KUNDU [24]	6.45 [6.37, 6.55]	3.85 [3.64, 4.01]	1.60 [1.58, 1.61]	1.51 [1.49, 1.53]	2.25 [2.23, 2.27]	2.09 [2.08, 2.10]
VIEN [40]	4.74 [4.37, 5.06]	2.02 [1.82, 2.21]	1.60 [1.57, 1.62]	1.54 [1.45, 1.58]	2.17 [2.16, 2.20]	2.15 [2.12, 2.17]
SHILTON [36]	6.66 [6.42, 7.04]	3.56 [3.45, 3.81]	1.39 [1.35, 1.41]	1.15 [1.07, 1.22]	2.15 [2.14, 2.20]	2.09 [2.07, 2.12]
VELLANKI [39]	✗	✗	✗	✗	2.36 [2.23, 2.93]	2.17 [2.14, 2.21]
L0MO	**2.70** [**2.65**, **2.81**]	**0.76** [**0.70**, **0.80**]	**0.66** [**0.61**, **0.84**]	**0.09** [**0.06**, **0.25**]	**2.11** [**2.10**, **2.12**]	**2.06** [**2.04**, **2.08**]

	Sinc-3D $[10^{-2}]$		**Hartmann** $[10^{0}]$		**Pendulum** $[10^{2}]$	
Method	**Avg**	**Final**	**Avg**	**Final**	**Avg**	**Final**
Random	5.22 [5.04, 5.41]	4.99 [4.76, 5.23]	3.14 [3.11, 3.16]	3.04 [3.00, 3.07]	5.29 [5.22, 5.31]	4.99 [4.94, 5.05]
KUNDU [24]	5.09 [5.00, 5.19]	3.59 [3.53, 3.63]	2.94 [2.91, 2.95]	2.65 [2.63, 2.68]	5.08 [5.05, 5.13]	4.91 [4.88, 4.93]
VIEN [40]	3.96 [3.78, 4.05]	3.16 [3.10, 3.25]	2.56 [2.54, 2.58]	2.37 [2.36, 2.40]	5.05 [5.01, 5.12]	4.63 [4.58, 4.72]
SHILTON [36]	5.46 [5.12, 5.68]	3.43 [3.40, 3.48]	2.58 [2.52, 2.59]	2.42 [2.38, 2.46]	4.74 [4.72, 4.77]	4.34 [4.30, 4.40]
VELLANKI [39]	✗	✗	✗	✗	✗	✗
L0MO	**3.40** [**3.36**, **3.44**]	**2.74** [**2.69**, **2.77**]	**1.36** [**1.35**, **1.37**]	**0.78** [**0.78**, **0.78**]	**4.22** [**4.12**, **4.33**]	**3.36** [**3.27**, **3.55**]

	Sinc-4D $[10^{-2}]$		**Rosenbrock** $[10^{4}]$		**Pinwheel** $[10^{0}]$	
Method	**Avg**	**Final**	**Avg**	**Final**	**Avg**	**Final**
Random	3.62 [3.58, 3.75]	3.46 [3.33, 3.60]	6.57 [6.51, 6.63]	6.10 [5.92, 6.21]	1.99 [1.87, 2.03]	1.23 [1.08, 1.42]
KUNDU [24]	4.39 [4.19, 4.54]	2.64 [2.62, 2.67]	5.91 [5.85, 5.94]	5.33 [5.31, 5.37]	1.75 [1.68, 1.83]	1.47 [1.42, 1.54]
VIEN [40]	2.73 [2.68, 2.76]	**2.43** [**2.42**, **2.44**]	5.39 [5.33, 5.49]	4.75 [4.68, 4.89]	2.95 [2.86, 3.08]	2.25 [2.19, 2.33]
SHILTON [36]	5.37 [4.79, 6.00]	2.60 [2.58, 2.63]	5.05 [4.97, 5.13]	4.42 [4.30, 4.46]	1.73 [1.69, 1.77]	1.50 [1.48, 1.59]
VELLANKI [39]	✗	✗	✗	✗	2.80 [2.79, 2.81]	2.53 [2.53, 2.54]
L0MO	**2.73** [**2.70**, **2.79**]	2.47 [2.46, 2.48]	**3.04** [**2.90**, **3.20**]	**0.20** [**0.19**, **0.22**]	**1.07** [**1.02**, **1.22**]	**0.12** [**0.10**, **0.18**]

	Sinc-5D $[10^{-2}]$		**Michalewicz** $[10^{0}]$		**Hopper** $[-10^{0}]$	
Method	**Avg**	**Final**	**Avg**	**Final**	**Avg**	**Final**
Random	4.59 [4.49, 4.72]	4.45 [4.26, 4.59]	4.20 [4.18, 4.21]	4.11 [4.09, 4.14]	1.84 [1.75, 1.95]	2.36 [2.14, 2.51]
KUNDU [24]	4.85 [4.75, 4.92]	3.62 [3.60, 3.66]	3.96 [3.93, 3.97]	3.72 [3.70, 3.75]	1.87 [1.79, 1.98]	1.92 [1.82, 2.02]
VIEN [40]	3.68 [3.64, 3.71]	**3.36** [**3.36**, **3.37**]	3.74 [3.72, 3.75]	3.38 [3.34, 3.42]	1.53 [1.50, 1.54]	1.59 [1.55, 1.63]
SHILTON [36]	6.55 [6.05, 6.96]	3.42 [3.40, 3.43]	3.81 [3.79, 3.82]	3.65 [3.63, 3.67]	2.23 [2.19, 2.28]	2.53 [2.43, 2.57]
VELLANKI [39]	✗	✗	✗	✗	1.46 [1.46, 1.46]	1.47 [1.47, 1.47]
L0MO	**3.67** [**3.63**, **3.70**]	3.40 [3.38, 3.43]	**3.22** [**3.21**, **3.22**]	**2.93** [**2.89**, **2.93**]	**2.36** [**2.28**, **2.53**]	**2.96** [**2.69**, **3.05**]

Limitations. Our analysis is not a convergence or regret guarantee: density of the sparse sets only ensures that good sparse approximations exist as m grows, and the acquisition landscape over the joint (α, x) space is highly non-convex, so multi-start L-BFGS-B, although effective in practice, may not find its global optimum. The method is best suited to objectives that can be approximately minimized by functions with few support points, and a smooth, non-sparse optimum such as the cycloid may be better served by dense bases, as the Brachistochrone results suggest. Our experiments use a complexity schedule shared across all methods, which some methods may not benefit from equally (section C), and a relatively coarse grid of RKHS lengthscales and GP kernels, so a finer search might reveal stronger configurations for one or more comparators. Finally, the cost of each acquisition grows with the number of support points and observations, which, as for any GP-based method, limits the approach to moderate evaluation budgets (section H).

We see several promising directions for future work. First, the complexity m is currently set by a fixed schedule; an adaptive schedule which selects m based on the data would be a natural extension. Second, our framework treats each acquisition independently; warm-starting the optimization from previously good candidates may further improve sample efficiency. Third, the natural gradient perspective opens up the use of other parameterizations, including those which enforce structural priors such as monotonicity or periodicity. Finally, the results on control problems suggest that FBO may be a viable strategy for policy optimization in low-data regimes; a more thorough study of this application is warranted.

References

[1] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[2] 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, 2024.

[3] J Betancourt, F Bachoc, T Klein, and J Rohmer. fungp: Gaussian process models for scalar and functional inputs, 2020.

[4] Mickaël Binois and Robert B Gramacy. hetgp: Heteroskedastic gaussian process modeling and sequential design in r. *Journal of Statistical Software*, 98:1–44, 2021.

[5] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[7] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[8] Luis Damiano, Margaret Johnson, Joaquim Teixeira, Max D Morris, and Jarad Niemi. Automatic dynamic relevance determination for gaussian process regression with high-dimensional functional inputs. *arXiv preprint arXiv:2209.00044*, 2022.

[9] 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.

[10] David Eriksson and Martin Jankowiak. High-dimensional Bayesian optimization with sparse axis-aligned subspaces. In Cassio de Campos and Marloes H. Maathuis, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 493–503. PMLR, 2021.

[11] Alexander I. J. Forrester, András Sóbester, and Andy J. Keane. Sampling plans. In *Engineering Design via Surrogate Modelling: A Practical Guide*, chapter 1. John Wiley & Sons, Ltd, 2008.

[12] Peter I. Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[13] Peter I. Frazier and Jialei Wang. Bayesian optimization for materials design. *arXiv preprint arXiv:1506.01349*, 2015.

[14] Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2023.

[15] Robert B Gramacy. *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences*. Chapman and Hall/CRC, 2020.

[16] Eric Han, Ishank Arora, and Jonathan Scarlett. High-dimensional Bayesian optimization via tree-structured additive models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7630–7638, 2021.

[17] James Hermus, Johannes Lachner, David Verdi, and Neville Hogan. Exploiting redundancy to facilitate physical interaction. *IEEE Transactions on Robotics*, 38(1):599–615, 2021.

[18] Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla Bayesian optimization performs great in high dimensions. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 20793–20817. PMLR, 2024.

[19] 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.

[20] Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*, 2018.

[21] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional Bayesian optimisation and bandits via additive models. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 295–304. PMLR, 2015.

[22] GilHwan Kim and Fabrizio Sergi. Validation of dynamic bayesian optimization for a non-stationary human-in-the-loop optimization problem. *IEEE Robotics and Automation Letters*, 2026.

[23] Johannes Kirschner, Mojmir Mutny, Nicole Hiller, Rasmus Ischebeck, and Andreas Krause. Adaptive and safe bayesian optimization in high dimensions via one-dimensional subspaces. In *International Conference on Machine Learning*, pages 3429–3438. PMLR, 2019.

[24] Poorbita Kundu and Nathan Wycoff. Active subspaces in infinite dimension. *arXiv preprint arXiv:2510.11871*, 2025.

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[26] Benjamin Letham, Roberto Calandra, Akshara Rai, and Eytan Bakshy. Re-examining linear embeddings for high-dimensional Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 33, pages 1546–1558, 2020.

[27] Thomas Muehlenstaedt, Jana Fruth, and Olivier Roustant. Computer experiments with functional inputs and scalar outputs by a norm-based approach. *Statistics and Computing*, 27(4):1083–1097, 2017.

[28] Amin Nayebi, Alexander Munteanu, and Matthias Poloczek. A framework for Bayesian optimization in embedded subspaces. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4752–4761. PMLR, 2019.

[29] Leonard Papenmeier, Luigi Nardi, and Matthias Poloczek. Increasing the scope as you learn: Adaptive Bayesian optimization in nested subspaces. In *Advances in Neural Information Processing Systems*, volume 35, pages 11586–11601. Curran Associates, Inc., 2022.

[30] Jack K. Pedersen, Christian M. Clausen, Olga A. Krysiak, Bin Xiao, Thomas A. A. Batchelor, Tobias Loffler, Vladislav A. Mints, Lars Banko, Matthias Arenz, Alan Savan, Wolfgang Schuhmann, Alfred Ludwig, and Jan Rossmeisl. Bayesian optimization of high-entropy alloy compositions for electrocatalytic oxygen reduction. *arXiv preprint arXiv:2106.08212*, 2021.

[31] Allan Pinkus. *Ridge functions*, volume 205. Cambridge University Press, 2015.

[32] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.

[33] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005.

[34] Paul Rolland, Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. High-dimensional Bayesian optimization via additive models with overlapping groups. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 84 of *Proceedings of Machine Learning Research*, pages 298–307. PMLR, 2018.

[35] Benjamin J Shields, Jason Stevens, Jun Li, Marvin Parasram, Farhan Damani, Jesus I Martinez Alvarado, Jacob M Janey, Ryan P Adams, and Abigail G Doyle. Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, 590(7844):89–96, 2021.

[36] Alistair Shilton, Sunil Gupta, Santu Rana, and Svetha Venkatesh. Sequential subspace search for functional bayesian optimization incorporating experimenter intuition. *arXiv preprint arXiv:2009.03543*, 2020.

[37] S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets, 2013. `http://www.sfu.ca/~ssurjano`, retrieved November 6, 2026.

[38] Milica Todorović, Michael U Gutmann, Jukka Corander, and Patrick Rinke. Bayesian inference of atomistic structure in functional materials. *Npj computational materials*, 5(1):35, 2019.

[39] Pratibha Vellanki, Santu Rana, Sunil Gupta, David Rubin de Celis Leal, Alessandra Sutti, Murray Height, and Svetha Venkatesh. Bayesian functional optimisation with shape prior. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1617–1624, 2019.

[40] Ngo Anh Vien, Heiko Zimmermann, and Marc Toussaint. Bayesian functional optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[41] Pascal Vincent and Yoshua Bengio. Kernel matching pursuit. *Machine learning*, 48(1):165–187, 2002.

[42] Ke Wang and Alexander W Dowling. Bayesian optimization for chemical products and functional materials. *Current Opinion in Chemical Engineering*, 36:100728, 2022.

[43] Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Freitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, 2016.

A RKHS Kernel Scale and Continuity Priors

Following the previous literature, we use an RKHS with an RBF kernel. The lengthscale parameter ℓ determines which functions admit a sparse approximation, and therefore changes the optimization landscape. Since the optimal lengthscale is task-specific rather than method-specific, we consider a log-uniform grid of values $\{0.4, 0.2, 0.1, 0.05\}$, shared by all methods on a benchmark. The selected lengthscale is the one at which the best configuration (method and GP profile) among the five compared methods attains its lowest median final value; ablation variants of L0MO do not take part in the selection.

The surrogate GP kernel also plays a role. The choice of kernel defines the prior on the smoothness of the surrogate model. We consider four profiles: RBF, Matérn-5/2, Matérn-3/2, and Matérn-1/2, corresponding respectively to $\mathcal{C}^\infty$, $\mathcal{C}^2$, $\mathcal{C}^1$, and $\mathcal{C}^0$ surrogate models. Unlike standard BO, where the target function is often assumed to be at least differentiable, in the functional domain the target may be discontinuous, making rougher kernels potentially more appropriate. The optimal profile may vary not only across target functions but also across methods, so we tune this independently for each method, showing each approach at its best. The selected lengthscales and profiles for each setting are reported in table 2.

Since the lengthscale selection could favor one method, table 3 reports the mean rank of each method over the ten main benchmarks at every lengthscale in the grid, and table 4 reports the underlying medians. L0MO has the best mean rank for both metrics at every lengthscale, and its advantage is largest at the longer lengthscales, where the sparse candidates are smoother.

Table 2: Selected hyperparameters per method and benchmark. The GP kernel determines the order of differentiability $\mathcal{C}(\nu)$ of the surrogate model ($\mathcal{C}(\infty)$ = RBF). The RKHS lengthscale ℓ is shared by all methods on a benchmark.

Benchmark	**RKHS scale**	**Surrogate kernel**			
		KUNDU [24]	VIEN [40]	SHILTON [36]	L0MO
GramacyLee	0.4	$\mathcal{C}(\infty)$	$\mathcal{C}(1)$	$\mathcal{C}(2)$	$\mathcal{C}(0)$
Ackley	0.4	$\mathcal{C}(2)$	$\mathcal{C}(0)$	$\mathcal{C}(1)$	$\mathcal{C}(2)$
Hartmann	0.4	$\mathcal{C}(1)$	$\mathcal{C}(\infty)$	$\mathcal{C}(\infty)$	$\mathcal{C}(2)$
Rosenbrock	0.4	$\mathcal{C}(2)$	$\mathcal{C}(\infty)$	$\mathcal{C}(\infty)$	$\mathcal{C}(1)$
Michalewicz	0.4	$\mathcal{C}(\infty)$	$\mathcal{C}(2)$	$\mathcal{C}(2)$	$\mathcal{C}(\infty)$
Brachistochrone	0.2	$\mathcal{C}(0)$	$\mathcal{C}(1)$	$\mathcal{C}(1)$	$\mathcal{C}(2)$
MNIST	0.4	$\mathcal{C}(0)$	$\mathcal{C}(1)$	$\mathcal{C}(0)$	$\mathcal{C}(0)$
Pendulum	0.2	$\mathcal{C}(1)$	$\mathcal{C}(0)$	$\mathcal{C}(2)$	$\mathcal{C}(1)$
Pinwheel	0.2	$\mathcal{C}(1)$	$\mathcal{C}(0)$	$\mathcal{C}(1)$	$\mathcal{C}(2)$
Hopper	0.1	$\mathcal{C}(0)$	$\mathcal{C}(\infty)$	$\mathcal{C}(\infty)$	$\mathcal{C}(1)$
Sinc-1D	0.05	$\mathcal{C}(\infty)$	$\mathcal{C}(2)$	$\mathcal{C}(\infty)$	$\mathcal{C}(1)$
Sinc-2D	0.1	$\mathcal{C}(\infty)$	$\mathcal{C}(\infty)$	$\mathcal{C}(\infty)$	$\mathcal{C}(\infty)$
Sinc-3D	0.2	$\mathcal{C}(1)$	$\mathcal{C}(2)$	$\mathcal{C}(1)$	$\mathcal{C}(\infty)$
Sinc-4D	0.4	$\mathcal{C}(1)$	$\mathcal{C}(2)$	$\mathcal{C}(0)$	$\mathcal{C}(\infty)$
Sinc-5D	0.4	$\mathcal{C}(1)$	$\mathcal{C}(\infty)$	$\mathcal{C}(0)$	$\mathcal{C}(\infty)$

Table 3: Mean rank over the ten main benchmarks at each RKHS lengthscale ℓ, each method using its own best GP profile at that ℓ. A method ties with any better one whose median lies within its own 95% CI. Lower is better; best per row in bold.

ℓ	Random		KUNDU [24]		VIEN [40]		SHILTON [36]		VELLANKI [39]		L0MO	
	Avg	**Final**	**Avg**	**Final**	**Avg**	**Final**	**Avg**	**Final**	**Avg**	**Final**	**Avg**	**Final**
0.05	4.40	4.70	3.70	3.20	2.80	2.30	2.20	2.90	3.20	3.20	**2.10**	**1.70**
0.1	4.20	4.40	3.40	3.30	3.10	2.90	2.60	2.60	3.20	3.20	**1.70**	**1.60**
0.2	4.30	4.40	3.80	3.60	3.10	3.40	2.20	2.50	3.40	3.40	**1.10**	**1.10**
0.4	3.50	3.80	4.10	4.20	3.10	3.50	1.90	2.40	4.60	4.00	**1.10**	**1.10**

Table 4: Median final and average regret at every RKHS lengthscale, each method at its own best GP profile. Each benchmark carries a common power of ten. Methods tied for first (see table 3) in bold.

Benchmark	Random		KUNDU [24]		VIEN [40]		SHILTON [36]		VELLANKI [39]		L0MO	
	Avg	**Final**	**Avg**	**Final**	**Avg**	**Final**	**Avg**	**Final**	**Avg**	**Final**	**Avg**	**Final**
RKHS lengthscale $\ell = 0.05$												
GramacyLee $[10^{-1}]$	7.30	6.37	6.53	6.37	6.90	6.37	6.46	6.37	**5.52**	**5.13**	6.89	6.37
Ackley $[10^{1}]$	1.69	1.68	1.69	1.68	**1.69**	1.68	**1.69**	**1.66**	✗	✗	**1.69**	1.68
Hartmann $[10^{0}]$	3.48	3.48	3.48	3.47	3.47	**3.46**	**3.47**	3.47	✗	✗	**3.47**	**3.46**
Rosenbrock $[10^{4}]$	7.87	7.86	7.86	7.86	**7.86**	**7.84**	**7.86**	7.85	✗	✗	7.86	7.84
Michalewicz $[10^{0}]$	4.37	4.37	4.37	4.37	4.37	**4.37**	**4.37**	4.37	✗	✗	4.37	**4.37**
Brachistochrone $[10^{-1}]$	2.66	2.51	3.03	2.16	1.86	1.15	3.53	2.83	0.91	0.23	**0.85**	**0.19**
MNIST $[10^{-2}]$	2.47	2.45	2.82	2.36	2.36	2.34	4.01	2.48	**2.34**	**2.17**	2.40	2.36
Pendulum $[10^{2}]$	5.48	5.42	5.34	5.21	5.37	5.23	5.31	5.20	✗	✗	**5.18**	**4.87**
Pinwheel $[10^{0}]$	1.82	1.17	**0.60**	**0.41**	1.95	1.64	**0.61**	0.57	2.80	2.53	1.78	1.16
Hopper $[-10^{0}]$	1.73	2.09	1.88	2.05	1.64	2.07	**2.16**	2.37	1.46	1.47	**2.12**	**2.43**
mean rank	4.40	4.70	3.70	3.20	2.80	2.30	2.20	2.90	3.20	3.20	**2.10**	**1.70**
RKHS lengthscale $\ell = 0.1$												
GramacyLee $[10^{-1}]$	6.38	5.41	6.57	6.37	5.89	5.13	5.79	5.13	**5.49**	**5.13**	6.48	6.37
Ackley $[10^{1}]$	1.68	1.65	1.62	**1.50**	1.69	1.67	**1.58**	**1.49**	✗	✗	**1.63**	**1.49**
Hartmann $[10^{0}]$	3.46	3.45	3.43	3.39	3.43	3.37	3.42	3.39	✗	✗	**3.34**	**3.20**
Rosenbrock $[10^{4}]$	7.81	7.77	7.73	7.65	7.69	7.54	7.69	7.60	✗	✗	**7.54**	**7.25**
Michalewicz $[10^{0}]$	4.37	4.37	4.36	4.36	4.36	4.36	4.36	4.36	✗	✗	**4.36**	**4.35**
Brachistochrone $[10^{-1}]$	2.04	1.83	1.79	0.74	1.24	0.64	2.28	1.39	0.91	0.23	**0.69**	**0.10**
MNIST $[10^{-2}]$	2.43	2.42	2.68	2.31	**2.31**	2.29	2.95	2.31	**2.42**	**2.17**	2.38	2.31
Pendulum $[10^{2}]$	5.36	5.21	5.09	4.80	5.16	4.78	4.99	4.77	✗	✗	**4.49**	**3.46**
Pinwheel $[10^{0}]$	1.75	**0.99**	**0.92**	**0.88**	2.25	1.90	1.10	1.01	2.80	2.53	1.75	0.94
Hopper $[-10^{0}]$	1.84	2.36	1.87	1.92	1.53	1.59	2.23	2.53	1.46	1.47	**2.36**	**2.96**
mean rank	4.20	4.40	3.40	3.30	3.10	2.90	2.60	2.60	3.20	3.20	**1.70**	**1.60**
RKHS lengthscale $\ell = 0.2$												
GramacyLee $[10^{-1}]$	6.08	5.33	6.48	6.37	6.34	5.18	**5.48**	**5.13**	**5.52**	**5.13**	**5.79**	**5.13**
Ackley $[10^{1}]$	1.64	1.57	1.56	1.43	1.60	1.49	**1.40**	**1.18**	✗	✗	1.60	**1.52**
Hartmann $[10^{0}]$	3.39	3.33	3.19	2.96	3.03	2.65	3.09	2.98	✗	✗	**2.27**	**1.06**
Rosenbrock $[10^{4}]$	7.48	7.24	6.76	6.04	6.64	5.57	6.67	6.24	✗	✗	**4.58**	**2.01**
Michalewicz $[10^{0}]$	4.34	4.32	4.28	4.21	4.28	4.16	4.27	4.23	✗	✗	**3.99**	**3.67**
Brachistochrone $[10^{-1}]$	1.24	0.98	1.08	0.52	0.74	0.33	**0.51**	**0.09**	0.91	0.23	**0.47**	0.10
MNIST $[10^{-2}]$	2.31	2.30	2.46	2.21	2.27	2.24	2.37	**2.16**	2.42	**2.17**	**2.22**	**2.16**
Pendulum $[10^{2}]$	5.29	4.99	5.08	4.91	5.05	4.63	4.74	4.34	✗	✗	**4.22**	**3.36**
Pinwheel $[10^{0}]$	1.99	1.23	1.75	1.47	2.95	2.25	1.73	1.50	2.80	2.53	**1.07**	**0.12**
Hopper $[-10^{0}]$	**1.64**	**1.87**	1.54	1.55	1.46	1.50	1.62	1.72	1.46	1.47	**1.69**	**1.77**
mean rank	4.30	4.40	3.80	3.60	3.10	3.40	2.20	2.50	3.40	3.40	**1.10**	**1.10**
RKHS lengthscale $\ell = 0.4$												
GramacyLee $[10^{-1}]$	5.53	5.19	6.56	6.37	5.45	5.13	5.50	5.13	5.52	5.13	**4.35**	**3.42**
Ackley $[10^{1}]$	1.58	1.46	1.60	1.51	1.60	1.54	1.39	1.15	✗	✗	**0.66**	**0.09**
Hartmann $[10^{0}]$	3.14	3.04	2.94	2.65	2.56	2.37	2.58	2.42	✗	✗	**1.36**	**0.78**
Rosenbrock $[10^{4}]$	6.57	6.10	5.91	5.33	5.39	4.75	5.05	4.42	✗	✗	**3.04**	**0.20**
Michalewicz $[10^{0}]$	4.20	4.11	3.96	3.72	3.74	3.38	3.81	3.65	✗	✗	**3.22**	**2.93**
Brachistochrone $[10^{-2}]$	7.65	3.64	8.68	7.06	7.16	5.41	**3.10**	**1.05**	9.15	2.26	4.22	1.25
MNIST $[10^{-2}]$	2.18	2.16	2.25	2.09	2.17	2.15	2.15	2.09	2.36	2.17	**2.11**	**2.06**
Pendulum $[10^{2}]$	5.27	5.01	5.27	5.17	5.17	5.05	**4.71**	**4.28**	✗	✗	**4.77**	**4.31**
Pinwheel $[10^{0}]$	2.34	1.88	2.33	1.99	3.04	2.69	2.26	2.05	2.80	2.53	**1.58**	**0.60**
Hopper $[-10^{0}]$	1.56	1.66	1.46	1.48	1.47	1.50	1.53	1.56	1.46	1.47	**1.67**	**1.76**
mean rank	3.50	3.80	4.10	4.20	3.10	3.50	1.90	2.40	4.60	4.00	**1.10**	**1.10**

B Effect of Natural Gradient

Figure 4 compares the optimization trajectories obtained when running the procedure of [40] with and without the natural gradient preconditioner. As established in section 4, the original updates of [40] correspond to natural gradient descent in a finite-dimensional parameter space; here we ablate the role of the preconditioner directly. The preconditioner is the pseudo-inverse G^{+} of the pullback metric, computed with the default singular value cutoff of the least squares solver.

The L-BFGS-B optimizer already builds an empirical approximation to the inverse Hessian from gradient history, which in practice subsumes much of the correction that the pullback metric would otherwise provide. For VIEN [40], the difference between running with and without the natural gradient preconditioner is small on most benchmarks, with a slight advantage for the natural gradient on Brachistochrone, Michalewicz and Pendulum.

For L0MO, where the parameterization jointly optimizes over support locations x and coefficients α, applying the natural gradient preconditioner on top of L-BFGS-B markedly *hurts* performance on GramacyLee, Ackley, Hartmann, Rosenbrock, Brachistochrone, Pendulum and Pinwheel, and

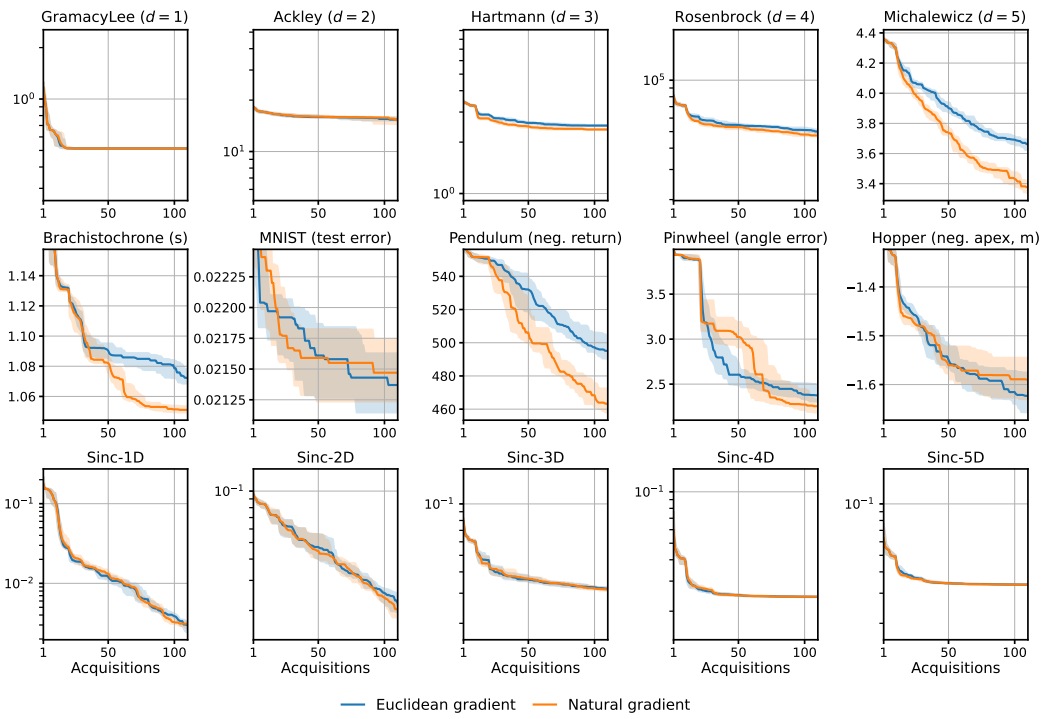

Figure 4: Effect of natural gradient preconditioning for VIEN [40] across benchmark functions.

slightly on Hopper; the two are comparable on the projection benchmarks and MNIST, and the natural gradient is better only on Michalewicz. Unlike for VIEN [40], whose support points are fixed during the acquisition optimization, the metric of L0MO depends on the support locations and degenerates when two of them approach each other or a coefficient approaches zero, which makes the preconditioned updates poorly scaled near these points. These results are consistent with quasi-Newton methods being effectively able to learn the proper metric such that explicitly using the preconditioner is superfluous. Since the ablations in sections C and E were run with the natural gradient variant of L0MO, they should be compared with it rather than with the results of section 5.

Regularization of the preconditioner. To test whether the degradation is caused by small singular values of G, fig. 5 compares the pseudo-inverse with truncated pseudo-inverses that discard singular values $s_i < r\, s_1$ for $r \in \{10^{-10}, 10^{-8}, 10^{-6}\}$. Truncation has no consistent effect beyond a small improvement on Brachistochrone: the variants remain close to each other, and all are worse than the Euclidean gradient on the benchmarks where the natural gradient hurts.

Symmetry of the parameterization. The parameterization (α, x) covers each sparse function $m!$ times, once per ordering of the support points (section 4). To test whether this symmetry hinders the optimizer, we restrict the search to a single sheet by parameterizing the first input coordinate of the support points as the normalized cumulative sum of positive gaps, which sorts the support points along that coordinate. Figure 6 shows that, on the ridge functionals and on Brachistochrone, Pendulum and Hopper, where it was run, the restriction has no appreciable effect with either gradient: the redundant sheets neither help nor hurt the multi-start optimization.

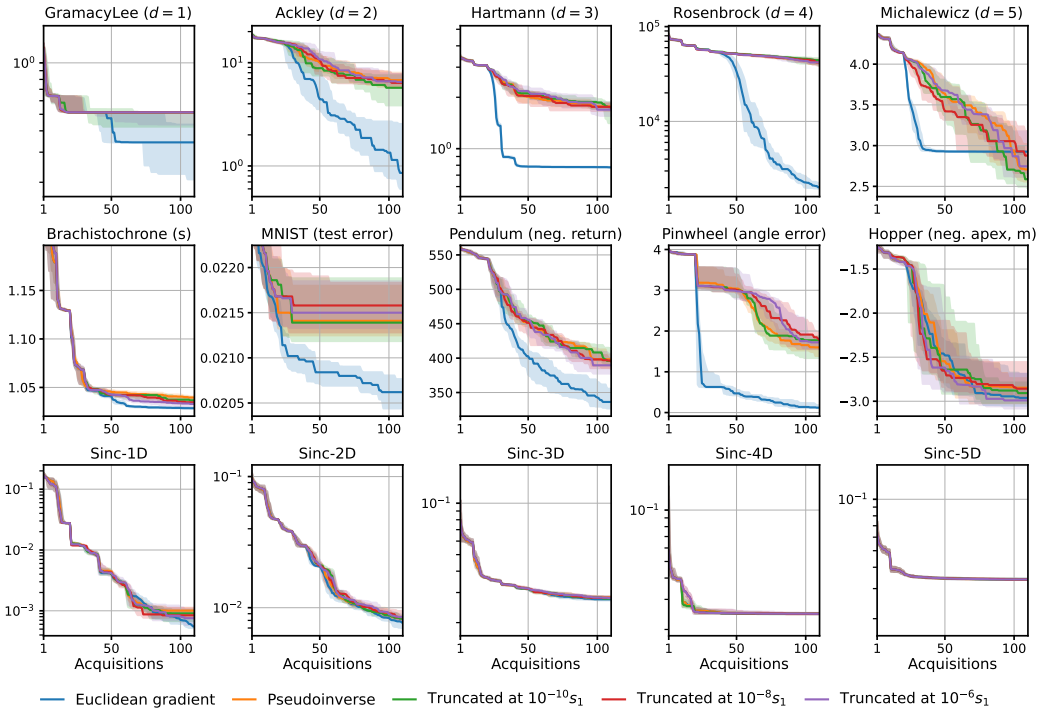

Figure 5: Effect of truncating the small singular values of the pullback metric in the natural gradient preconditioner of L0MO, compared with the pseudo-inverse and the Euclidean gradient.

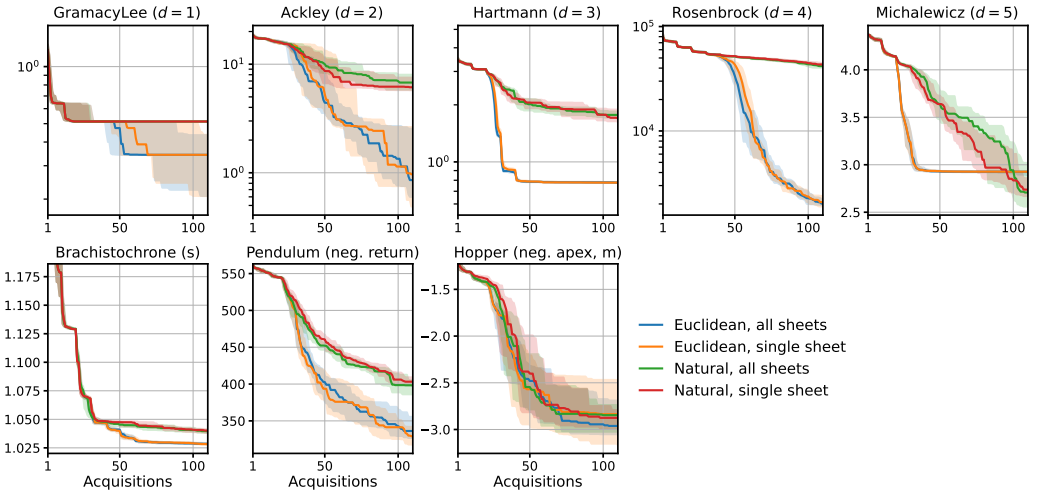

Figure 6: Effect of restricting L0MO to a single sheet of the parameterization, with Euclidean and natural gradients, on the benchmarks where the single-sheet variant was run.

C Complexity Schedule

All methods in section 5 share a schedule in which the complexity k grows from 1 to 10 every 10 acquisitions. Figure 7 compares this schedule with keeping k fixed at 1, 5 or 10 throughout the run, for L0MO (natural gradient variant). A single support point ($k = 1$) is the worst choice on nearly every benchmark, with the exceptions of MNIST, Sinc-4D and Sinc-5D, whose optima are well approximated by very simple functions. Fixing $k = 5$ or $k = 10$ improves over the growing schedule on several ridge functionals and on the Pendulum, Pinwheel and Hopper tasks, but is worse on

Brachistochrone and clearly worse on MNIST and the projection benchmarks. No fixed complexity dominates, and the growing schedule is the most robust choice across benchmarks; an adaptive schedule that selects k from the data is a natural direction for future work.

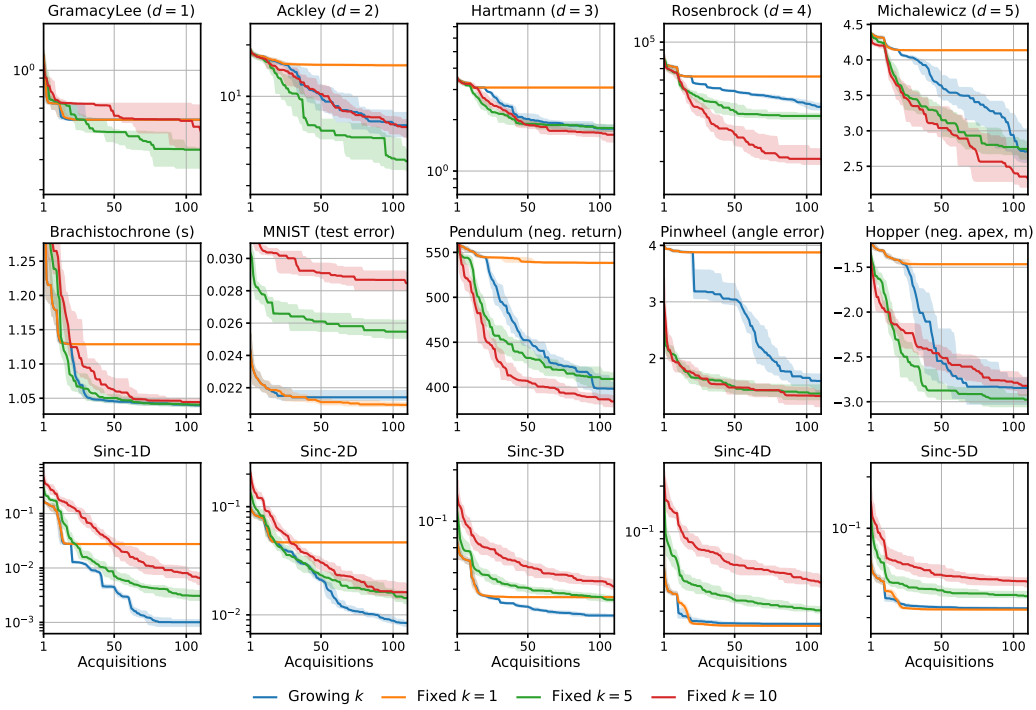

Figure 7: Growing complexity schedule compared with a fixed number of support points k for L0MO.

D Gaussian Process Kernel Profile

The profile of the surrogate GP kernel determines the smoothness assumptions placed on the target functional f. In standard finite-dimensional BO, the target is commonly assumed to be at least once differentiable, and the RBF or Matérn-5/2 kernel is a natural default. In the functional domain, however, f maps functions to scalars and need not inherit the regularity of the underlying function space; for instance, evaluation-based objectives such as travel time or cumulative reward can be discontinuous as a function of the RKHS distance between candidates. This motivates considering rougher kernels, including Matérn-3/2 and Matérn-1/2, as surrogate priors.

Figure 8 shows the performance of L0MO under each of the four kernel profiles across all benchmarks, at the selected RKHS lengthscale. The profile matters substantially on Ackley, Rosenbrock, Pinwheel and Sinc-1D, where the median final value varies by an order of magnitude across profiles, and hardly at all on Hartmann, Michalewicz, MNIST, Sinc-4D and Sinc-5D. Matérn-5/2 and Matérn-3/2 are the best or close to the best on most benchmarks, while the roughest profile, Matérn-1/2, is the worst on most synthetic benchmarks and on Brachistochrone and Pinwheel, but the best on GramacyLee and MNIST. No single profile dominates uniformly, and the selected profile also varies across methods on the same benchmark (table 2). In the main experiments of section 5, each method is therefore evaluated at the profile that achieves the lowest median final value for that method and benchmark, so that no method is penalized by a suboptimal kernel choice. Table 5 shows that the conclusions of section 5 do not depend on this per-method selection: with the Matérn-5/2 profile for every method, L0MO is tied for first on every benchmark and metric except the final value on Brachistochrone, where SHILTON [36] is marginally better.

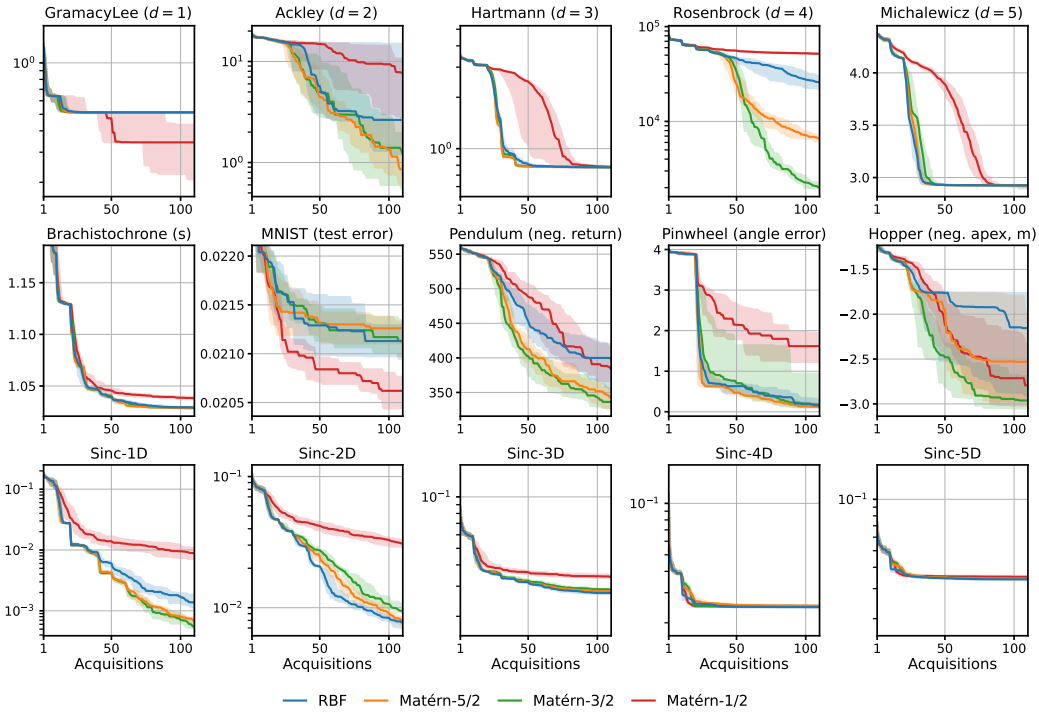

Figure 8: Effect of the GP kernel profile on L0MO across benchmark functions. Each line corresponds to a different smoothness assumption on the surrogate.

Table 5: Median final and average regret with the Matérn-5/2 GP profile for every method, at the selected RKHS lengthscale. Methods tied for first in bold.

Benchmark	Random		KUNDU [24]		VIEN [40]		SHILTON [36]		VELLANKI [39]		L0MO	
	Avg	**Final**	**Avg**	**Final**	**Avg**	**Final**	**Avg**	**Final**	**Avg**	**Final**	**Avg**	**Final**
GramacyLee [10^{-1}]	**5.53**	5.19	6.58	6.37	**5.45**	5.13	**5.50**	**5.13**	5.46	**5.13**	**5.39**	**5.13**
Ackley [10^{1}]	1.58	1.46	1.60	1.51	1.61	1.58	1.39	1.16	✗	✗	**0.66**	**0.09**
Hartmann [10^{0}]	3.14	3.04	2.94	2.65	2.55	2.38	2.58	2.43	✗	✗	**1.36**	**0.78**
Rosenbrock [10^{4}]	6.57	6.10	5.91	5.33	5.51	5.10	5.07	4.43	✗	✗	**3.17**	**0.66**
Michalewicz [10^{0}]	4.20	4.11	3.96	3.72	3.74	3.38	3.81	3.65	✗	✗	**3.24**	**2.93**
Brachistochrone [10^{-1}]	1.24	0.98	1.21	0.74	0.74	0.38	**0.52**	**0.09**	0.91	0.23	**0.47**	0.10
MNIST [10^{-2}]	2.18	2.16	2.28	2.19	2.18	2.15	2.18	**2.14**	2.34	2.18	**2.15**	**2.13**
Pendulum [10^{2}]	5.29	4.99	5.09	4.95	5.15	4.75	4.74	4.34	✗	✗	**4.31**	**3.43**
Pinwheel [10^{0}]	1.95	1.26	1.79	1.55	3.17	2.78	1.72	1.54	2.82	2.54	**1.07**	**0.12**
Hopper [-10^{0}]	1.84	2.36	1.86	1.89	1.50	1.54	**2.23**	**2.51**	1.46	1.47	**2.16**	**2.53**
mean rank	3.80	4.00	4.00	4.30	3.10	3.50	1.80	1.80	4.40	3.80	**1.00**	**1.10**

E Initial Candidate Distribution

At each acquisition step, the L-BFGS-B optimizer is initialized from a set of 16 candidate starting points, selected as the top 16 out of 1024 initial guesses evaluated on the acquisition function. The default strategy samples these 1024 guesses via Latin Hypercube Sampling (LHS) in the parameter space, as described in section F. Following [36], an alternative is to draw the initial guesses from the surrogate GP prior.

Figure 9 compares these two sampling strategies for L0MO (natural gradient variant) across all benchmarks, overlayed with SHILTON [36] for reference. In the early acquisitions, the choice of initial candidate distribution produces a visible difference: prior-based sampling can lead to better or worse starting points depending on how suitable the GP prior is for the given test function. As the number of acquisitions grows, the two strategies reach similar final values on most benchmarks, with differences of no consistent sign: prior-based starts are better on Hartmann and Brachistochrone, and worse on Michalewicz and, in average regret, on Hopper. This is consistent with the expectation that,

given a sufficiently good acquisition optimizer, the final solution found by L-BFGS-B is determined mostly by the acquisition landscape rather than by the initialization, and the top-16 selection already filters out poor starting points effectively. These results suggest that the initial candidate distribution is not a critical design choice for FBO.

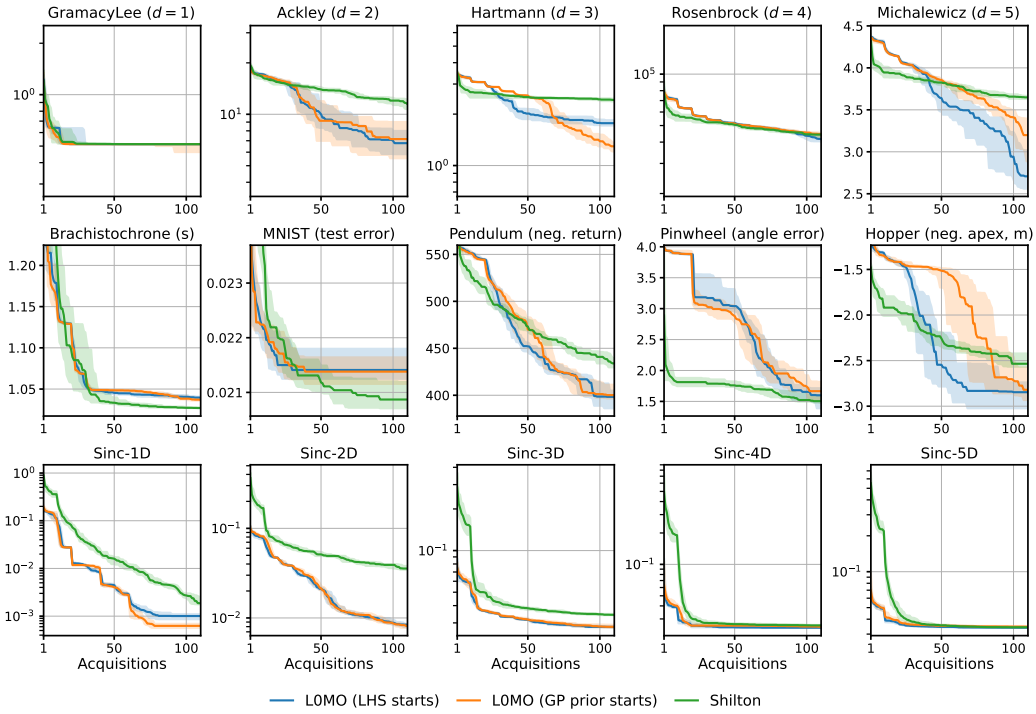

Figure 9: Effect of the initial candidate distribution across benchmark functions.

F Implementation Details

The surrogate GP is fit by maximum likelihood estimation of the lengthscale ℓ and nugget g, with the optimal trend b and scale ν estimated analytically. For L0MO, VIEN [40], KUNDU [24], and SHILTON [36], the surrogate takes functional inputs u with a stationary kernel defined on RKHS distances $\|u - u'\|_{\mathcal{H}}$. For VELLANKI [39], the surrogate works directly on the Bernstein polynomial coefficients, bypassing the RKHS representation entirely. Following HETGP [4], the lengthscale bounds are set from the 5% and 95% quantiles of the pairwise distances between observations, so that the closest pairs can decorrelate down to a correlation of 0.01 and the farthest pairs can remain correlated at 0.5; the nugget is bounded in $[\varepsilon_{\text{machine}}, 100]$. The observations are zero-padded and masked to the next power of two, so that the likelihood and the predictive equations are compiled once per padded size rather than once per acquisition, and the LU factorization of the covariance matrix is computed once per fit and reused throughout the acquisition optimization.

The acquisition function used follows the numerically stable log-EI implementation of [2]. All methods use the same acquisition optimization strategy: given 1024 initial candidates sampled via LHS, the acquisition loss is evaluated at each candidate and the top 16 are selected as starting points for L-BFGS-B optimization with `maxiter`$= 100$, `ftol`$= \sqrt{\varepsilon_{\text{machine}}}$, and `gtol`$= 0$. The best result across restarts is returned.

Since the surrogate GP operates on RKHS distances, it can be fit on functional inputs with varying numbers of support points. Therefore the complexity of candidate points can vary across acquisition steps. Finding the optimal complexity schedule is beyond the scope of this work (see section C for a comparison with fixed complexities). For our evaluation we adopt a fixed schedule, shared across all methods. The parameter k is taken to represent the size of the acquisition optimization problem: k is the subspace dimension for KUNDU [24] and SHILTON [36], the number of support points for

VIEN [40] and L0MO, and the polynomial degree for VELLANKI [39]. We evaluate all methods on a fixed budget of 110 function evaluations, beginning with 10 initial candidates sampled via LHS, followed by 10 sequential acquisitions at each complexity level $k \in \{1, \ldots, 10\}$. The RANDOM baseline follows the same schedule but skips surrogate modeling, using LHS for all acquisitions.

The RKHS lengthscale is selected per benchmark from $\{0.4, 0.2, 0.1, 0.05\}$ and the GP kernel profile per method and benchmark, as detailed in sections A and D. Each configuration (benchmark, method, kernel profile and lengthscale) is repeated with 16 to 32 independent seeds, for a total of about 105k individual runs. The sweep was carried out on a Slurm cluster, and we provide a script to reproduce it, including preset time and memory requirements. Each individual run can be reproduced on a consumer laptop, with running times ranging from a few minutes for the synthetic benchmarks to a few hours for neural network training and control simulations (section H).

All benchmarks operate in the same function space, optimizing over candidate functions $u : [0, 1]^d \to [-1, 1]$. To avoid discrepancies arising from differing input-output domains, additional care is needed to homogenize representation across methods.

VIEN [40]. Initial candidates are sampled via the distribution induced by LHS and the bounded parameterization with $k_{\min}$ support points. At each acquisition step, new candidate functions are first sampled via LHS with k support points. Each candidate is then expanded to include the union of all historical observation support points $\{x_i\}$, with zero coefficients appended for the new points. The function values at all expanded support points are soft-squished to $[-1, 1]$ using a scaled logistic function. Given this final step, the induced metric is no longer exactly given by the canonical RKHS inner product. We compute it using the general pullback formula. After optimization, KMP [41] is applied to sparsify the result to exactly k support points before target evaluation.

KUNDU [24]. The basis functions for the random subspace are sampled with $k_{\max}$ support points throughout. At each acquisition step a fresh random subspace of dimension k is constructed by sampling k basis functions via the distribution induced by LHS and the bounded parameterization. The candidate is a linear combination of these basis functions with coefficients $c \in [-1, 1]$, obtained by concatenating the support points and coefficients of all basis functions. The corresponding y values are computed and soft-squished to enforce the $[-1, 1]$ range. The final candidate is reconstructed using the squished values.

SHILTON [36]. At each acquisition step, a fresh random subspace of dimension k is constructed by sampling k basis functions from the surrogate GP prior on a set of $n = 50$ points in $[0, 1]^d$ drawn via LHS. The original paper suggests using $n = 100$, but we found that denser grids led to numerical instability. The subspace is centered at the incumbent, i.e. the observed function u_n with the lowest target value $f(u_n)$ among $u_1, \ldots, u_N$. The y values of the linear combination at the support points are soft-squished to $[-1, 1]$, and the candidate is reconstructed via the bounded parameterization and sparsified to $k_{\max}$ support points with KMP before target evaluation.

VELLANKI [39]. Candidates are represented as Bernstein polynomials of degree k with coefficients $c \in [-1, 1]^{k+1}$, sampled via LHS and rescaled to $[-1, 1]^{k+1}$. At evaluation time, the input is soft-squished to map to $[0, 1]$ in order to make the method usable on the one-dimensional benchmarks. When the complexity level k increases, all existing observations are lifted to degree k via the exact Bernstein degree elevation recurrence: $c_i^{(n+1)} = \frac{i}{n+1} c_{i-1}^{(n)} + \left(1 - \frac{i}{n+1}\right) c_i^{(n)}$. The surrogate GP is fit from scratch on the lifted coefficients. This method only supports $d = 1$.

G Convergence of the Integral Functionals

The ridge functionals of section 5 are defined by the integrals $L_i[u] = \int w_i u$. For candidates in the RBF RKHS, each integral factorizes over the input dimensions and each factor is a Gaussian integral of a trigonometric function over $[0, 1]$, which we evaluate in closed form through the complex error function. To check this closed form, and to show how accurately the functional can be approximated when it is not available, we compare it with the Monte Carlo estimate obtained by averaging $w_i u$ over n uniform points in $[0, 1]^d$. We draw 32 random candidates with 10 support points and lengthscale 0.2, in the same way as the initial designs of the sweep, and evaluate the ratio between the Monte

Carlo and closed form target values for $n = 2^0, \ldots, 2^{16}$. Figure 10 shows that the ratio converges to one on every benchmark: the median relative error is below 1.3% at $n = 2^{12}$ and below 0.3% at $n = 2^{16}$, decreasing at the expected $O(1/\sqrt{n})$ rate.

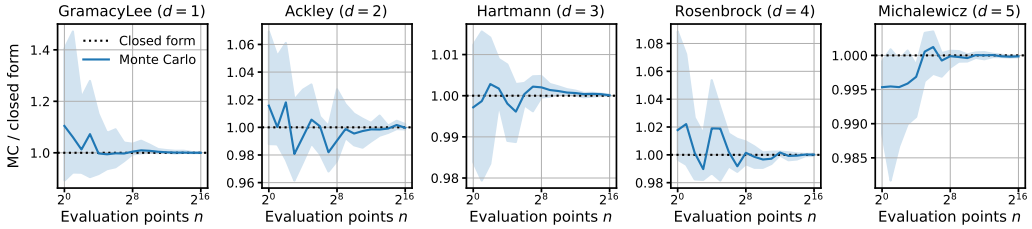

Figure 10: Ratio between the Monte Carlo estimate of the ridge functionals on n uniform points and their closed form value (median and 95% CI over 32 random candidates).

H Wall-Clock Time

We report two timing studies. Table 6 reports the total time per run spent fitting the surrogate and optimizing the acquisitions, measured on a single CPU core so that all methods run on the same hardware, and averaged across benchmarks. L0MO sits in the middle of the pack: it is cheaper than VIEN [40], whose candidates carry the support points of all previous observations, and than the subspace methods KUNDU [24] and SHILTON [36] in this setting, while VELLANKI [39] has the cheapest acquisition since it optimizes $k + 1$ polynomial coefficients only. The surrogate fit costs about the same for all RKHS-based methods, since they share the same GP on RKHS distances.

Table 7 breaks the times down per benchmark, as recorded during the main sweep. Since the sweep ran on a heterogeneous cluster, these times depend on the node each run was scheduled on and are comparable only within a row. The acquisition cost of L0MO grows with the number of optimized parameters $k(d+m)$, and is largest on Hopper, where the policy has two outputs, and on the expensive MNIST and Pinwheel benchmarks.

Table 6: Total wall-clock time per run (seconds) spent fitting the surrogate and optimizing the acquisition, over the 110-evaluation budget, on a single CPU core and averaged across benchmarks.

Time (s)	Random	KUNDU [24]	VIEN [40]	SHILTON [36]	VELLANKI [39]	L0MO
Surrogate fit	✗	85.5	88.5	81.1	154.1	86.9
Acquisition	3.9	254.9	586.1	891.7	79.7	202.2

Table 7: Median total wall-clock time per run (seconds) spent fitting the surrogate and optimizing the acquisition, over the 110-evaluation budget, as recorded in the main sweep on a heterogeneous cluster. Times include JAX compilation and are only indicative across rows.

Benchmark	KUNDU [24]		VIEN [40]		SHILTON [36]		VELLANKI [39]		L0MO	
	Fit	**Acq.**	**Fit**	**Acq.**	**Fit**	**Acq.**	**Fit**	**Acq.**	**Fit**	**Acq.**
GramacyLee	66	191	58	496	49	71	108	72	86	78
Ackley	87	171	63	455	45	30	✗	✗	99	102
Hartmann	69	120	69	437	67	42	✗	✗	82	89
Rosenbrock	96	174	74	503	72	36	✗	✗	98	129
Michalewicz	67	173	88	756	80	46	✗	✗	92	234
Brachistochrone	75	291	86	563	438	1262	160	70	90	134
MNIST	15362	1392	66	386	442	1266	5469	175	1320	1671
Pendulum	83	182	99	560	79	31	✗	✗	94	143
Pinwheel	2040	1155	81	509	424	1116	820	205	1246	1487
Hopper	81	621	102	1238	105	671	165	79	94	1247
Sinc-1D	88	213	76	559	104	59	177	67	84	126
Sinc-2D	91	160	103	522	96	52	✗	✗	104	121
Sinc-3D	106	189	97	460	106	56	✗	✗	63	109
Sinc-4D	82	183	101	623	89	62	✗	✗	67	137
Sinc-5D	75	152	102	570	86	162	✗	✗	91	156

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We mention our methodology and experimental claims in abstract and substantiate.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The Limitations paragraph of section 6 covers the lack of convergence or regret guarantees, the non-convex acquisition problem, the restriction to objectives that are well approximated by few support points, the shared complexity schedule, the coarse hyperparameter grid and the computational cost, with pointers to the corresponding appendices.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper is primarily methodological. Its formal statements (the structure of the sparse set as a finite union of immersed manifolds, density of its union in $\mathcal{H}$, existence of a maximizer of the acquisition problem, and the equivalence of the updates of prior work with natural gradient descent) are stated with their assumptions and derived in section 4. We do not claim convergence or regret guarantees.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We give details of numerical experiments in Section 5 and in appendices.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code, including the benchmark implementations and the script reproducing the full experimental sweep, is available at `https://github.com/davidesartor/bofus`. All data used are either generated by the benchmarks or publicly available (MNIST).

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes we discuss this in the experimental section and in the appendices.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All figures and tables report the median over 16 to 32 independent seeds together with a 95% confidence interval for the median, computed from the order statistics at quantiles $0.5 \pm 1.96\sqrt{0.25/n}$ (normal approximation to the binomial), which does not assume normally distributed outcomes.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section F describes the sweep (about 105k CPU runs on a Slurm cluster) and section H reports wall-clock times per run and per method, both on a single CPU core and per benchmark as recorded during the sweep. The repository provides the scripts to reproduce it with preset time and memory requirements.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We followed that code.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: This is foundational research on optimization algorithms, not tied to particular applications, and we do not foresee a direct societal impact beyond that of black-box optimization in general.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: Not applicable to this kind of research.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We re-implemented the methods of prior work from scratch and cite them. The existing assets we use are cited: MNIST [25] (CC BY-SA 3.0), the Pendulum environment of Gym/Gymnasium [6] (MIT license), JAX [5] (Apache 2.0), and the benchmark functions of the virtual library of simulation experiments [37], which we re-implemented from their published definitions.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Our code is released at `https://github.com/davidesartor/bofus` under the MIT license, with a README documenting installation, the methods and the benchmarks.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: No crowdsourcing or human subjects research performed.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human research conducted.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [No]

    Justification: LLMs were not used for the core methods of research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.