

Nonlinear dimension reduction for surrogate modeling using gradient information

Daniele Bigoni, Youssef Marzouk, Clémentine Prieur, Olivier Zahm

▶ **To cite this version:**

Daniele Bigoni, Youssef Marzouk, Clémentine Prieur, Olivier Zahm. Nonlinear dimension reduction for surrogate modeling using gradient information. Information and Inference, 2022. hal-03146362

HAL Id: hal-03146362
https://inria.hal.science/hal-03146362

Submitted on 19 Feb 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Nonlinear dimension reduction for surrogate modeling using gradient information

Daniele Bigoni[*], Youssef Marzouk[*], Clémentine Prieur[†] and Olivier Zahm[†‡]

February 20, 2021

Abstract

We introduce a method for the nonlinear dimension reduction of a high-dimensional function $u : \mathbb{R}^d \to \mathbb{R}$, $d \gg 1$. Our objective is to identify a nonlinear feature map $g : \mathbb{R}^d \to \mathbb{R}^m$, with a prescribed intermediate dimension $m \ll d$, so that u can be well approximated by $f \circ g$ for some profile function $f : \mathbb{R}^m \to \mathbb{R}$. We propose to build the feature map by aligning the Jacobian ∇g with the gradient ∇u, and we theoretically analyze the properties of the resulting g. Once g is built, we construct f by solving a gradient-enhanced least squares problem. Our practical algorithm makes use of a sample $\{\boldsymbol{x}^{(i)}, u(\boldsymbol{x}^{(i)}), \nabla u(\boldsymbol{x}^{(i)})\}_{i=1}^N$ and builds both g and f on adaptive downward-closed polynomial spaces, using cross validation to avoid overfitting. We numerically evaluate the performance of our algorithm across different benchmarks, and explore the impact of the intermediate dimension m. We show that building a nonlinear feature map g can permit more accurate approximation of u than a linear g, for the same input data set.

Keywords high-dimensional approximation, nonlinear dimension reduction, feature map, Poincaré inequality, adaptive polynomial approximation.

1 Introduction

Computational models from a wide range of fields, such as physics, biology, and finance, involve large numbers of uncertain input parameters. Quantifying uncertainty is essential to improving the reliability of these models. Most uncertainty quantification analyses, however, require a large number of model evaluations. When a single evaluation is computationally expensive, a common practice is therefore to replace the model with a *surrogate*—meaning an approximation that can be evaluated cheaply, without further evaluations of the original model. Yet constructing accurate

[*]Center for Computational Science & Engineering, Massachusetts Institute of Technology
[†]Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France
[‡]Corresponding author (olivier.zahm@inria.fr)

approximations is a challenging task because many function approximation tools become inexpressive in high dimensions. This is often referred as to the *curse of dimensionality*. This problem is exacerbated in the small-data regime, i.e., when few model evaluations are available.

This paper addresses the problem of reducing parameter space dimension from the perspective of surrogate modeling. We represent the model by a scalar-valued quantity of interest $u(\boldsymbol{x})$ which depends on a high dimensional parameter $\boldsymbol{x} \in \mathbb{R}^d$ with $d \gg 1$. When the parameter is uncertain, it is denoted by a random vector $\boldsymbol{X}$ whose law models the uncertainty of the parameter. Dimension reduction consists in finding a map $g : \mathbb{R}^d \to \mathbb{R}^m$, with $m \ll d$, that captures the most "relevant" features of the parameters. This *feature map* permits reduction of the parameter dimension from d to m by replacing $\boldsymbol{X}$ with the m-dimensional random vector $\boldsymbol{Z} = g(\boldsymbol{X})$. From the perspective of surrogate modeling, a good feature map should enable $u(\boldsymbol{X})$ to be well approximated as $f(\boldsymbol{Z}) = f \circ g(\boldsymbol{X})$, for some function $f : \mathbb{R}^m \to \mathbb{R}$ of m variables only. If such a feature map g is known in advance, f can be constructed by minimizing the mean squared error,

$$\mathbb{E}\left[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2\right],$$

over a class of functions of $m \ll d$ variables. This task is, in principle, easier than constructing a d-dimensional approximation to $u(\boldsymbol{X})$ directly.

Linear dimension reduction corresponds to identifying linear feature maps g. Many linear dimension reduction strategies have been proposed in different research fields. Global sensitivity analysis [36] identifies a set of m parameters $g(\boldsymbol{x}) = (x_{\sigma_1}, \ldots, x_{\sigma_m})$ that best explain, in some statistical sense, the model output. More generally, ridge functions [34] are functions of the form $\boldsymbol{x} \mapsto f \circ g(\boldsymbol{x})$ where $g(\boldsymbol{x}) = W^T\boldsymbol{x}$ for some matrix $W \in \mathbb{R}^{d\times m}$. In [9, 16], the model u is assumed to be a ridge function and W is recovered via adaptive model query strategies. Linear dimension reduction also arises in the statistical regression literature under the name *sufficient dimension reduction* [2, 28], where W is constructed via sliced inverse regression (SIR) [29], sliced average variance estimation (SAVE) [13], and their variants. Closely related to the present work is the active subspace method [12, 11, 23], which identifies W using gradients of the model. The recent papers [45, 32] show that the active subspace method constructs the matrix W by minimizing an upper bound for the mean squared error $\mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2]$ optained with the optimal profile function. This result is particularly relevant because it motivates the construction of g from the perspective of approximating u in the least-squares sense. Similar ideas are developped in [46, 14, 7] for the detection of informed subspace in the context of Bayesian inverse problems.

While linear dimension reduction methods are quite successful in many applications, they can fail to detect certain kinds of low-dimensional structure that a model might have; consider an isotropic $u(\boldsymbol{x}) = h(\|\boldsymbol{x}\|)$, for instance. *Nonlinear dimension reduction* allows g to detect such nonlinear features, in order to improve the approximation power of the composed approximation $f \circ g$. Nonlinear dimension reduction methods have been developed and analyzed mostly in the community of sufficient dimension reduction; see for instance [43, 44, 27], to cite just a few. In these works, the main idea is to use kernel methods to construct a nonlinear feature map $g(\boldsymbol{x}) = W^T\Phi(\boldsymbol{x})$, where $\Phi(\boldsymbol{x}) = (\Phi_1(\boldsymbol{x}), \Phi_2(\boldsymbol{x}), \ldots)$ are the eigenfunctions of an ad hoc kernel (typically a squared exponential or a polynomial kernel), and where the matrix W is determined using inverse regression techniques (SIR, SAVE) on the transformed variables $\boldsymbol{Z} = \Phi(\boldsymbol{X})$. Those methods, however, typically require a large sample size to accurately detect the low-dimensional structure of the model, and thus are not well suited to the small-data regime. In the spirit of kernel principal component analysis (KPCA), [25] builds a feature map of the form $g(\boldsymbol{x}) = (\Phi_1(\boldsymbol{x}), \ldots, \Phi_m(\boldsymbol{x}))$ by taking the m first eigenfunctions of a kernel whose hyperparameters (e.g., correlation length, smoothness) are determined by an outer optimization procedure.

1.1 Contribution

The main contribution of this paper is to propose and analyze a nonlinear parameter space dimension reduction method, for the purpose of function approximation, using gradients of the model. We assume here that the implementation of the computational model permits computing the gradient of $\boldsymbol{x} \mapsto u(\boldsymbol{x})$ with respect to the parameters $\boldsymbol{x}$. Recent advances in computational science permit computing such gradients at a complexity comparable to that of evaluating the model itself, for instance using automatic differentiation [19] and/or adjoint state methods [35]. Having access to gradient evaluations is a valuable workaround in small-data regimes, as $\nabla u(\boldsymbol{X})$ constitutes additional information for learning the model; see [26]. In this paper we propose to build g by minimizing the loss function

$$J(g) = \mathbb{E}\left[\left\|\nabla u(\boldsymbol{X}) - \Pi_{\mathrm{range}(\nabla g(\boldsymbol{X})^T)} \nabla u(\boldsymbol{X})\right\|^2\right],$$

where $\Pi_{\mathrm{range}(\nabla g(\boldsymbol{X})^T)}$ denotes the orthogonal projector onto the range of the Jacobian $\nabla g(\boldsymbol{X})^T$. Intuitively, minimizing this loss yields a feature map whose Jacobian $\nabla g(\boldsymbol{X})$ tends to be aligned with the gradient $\nabla u(\boldsymbol{X})$. Based on the same heuristic, the authors of [47] introduce a different loss function to align $\nabla g(\boldsymbol{X})$ with $\nabla u(\boldsymbol{X})$ (see Appendix A for more details) but without proposing a deeper mathematical or computational analysis. In the present paper, we prove that, under some assumptions, the loss $J(g)$ yields an upper bound on the mean squared error that can be obtained after constructing f; that is

$$\min_{f:\mathbb{R}^m \to \mathbb{R}} \mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2] \leq \mathbb{C}\ J(g),$$

for some Poincaré-type constant $\mathbb{C}$ associated with $\boldsymbol{X}$. We propose a quasi-Newton algorithm to minimize $J(g)$ and show that this algorithm is similar to the power iteration used to compute an eigendecomposition in the active subspace method.

In practice, we make use of a data set

$$\{\boldsymbol{x}^{(i)}, u(\boldsymbol{x}^{(i)}), \nabla u(\boldsymbol{x}^{(i)})\}_{i=1}^N,$$

to estimate the loss $J(g)$ and the mean squared error $\mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2]$. We assume that the computational cost is dominated by the N evaluations of $u(\boldsymbol{x}^{(i)})$ and $\nabla u(\boldsymbol{x}^{(i)})$, such that the cost for constructing f and g is relatively negligible. Borrowing ideas from [5, 30, 10], we represent both f and g on adaptive downward-closed polynomial spaces which are built using a greedy algorithm. In order to avoid overfitting, a cross validation procedure is used to determine when to stop the adaptive polynomial enrichment. We show that building a nonlinear feature map g permits more accurate approximation of u than a linear g, for the same input data set.

We emphasize that our method is a two step procedure: we first build the feature map g by minimizing $J(g)$, and we then build f by minimizing the mean squared error $\mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2]$. Another strategy would consist of minimizing the mean squared error *jointly* over f and g. For instance, in [20] the authors build a linear g and polynomial f by employing dedicated optimization algorithms on Grassmann manifolds, without using gradients of the model. Nonlinear g are also built in [25] by joint minimization over f and g. However, the structure of such optimization problems, and of the algorithms they employ, remain not well understood.

The rest of this paper is organized as follows. In Section 2 we analyze the problem of approximating a function u by a composition $f \circ g$. In particular, we give sufficient conditions on ∇g and ∇u so that there exists an f such that $f \circ g = u$. We then introduce the loss $J(g)$ and describe its properties regarding the approximation problem. In Section 3 we present algorithms for constructing g and f on adaptive polynomial spaces. Then, in Section 4, we illustrate the method on numerical examples.

2 Dimension reduction via smooth feature maps

2.1 Problem statement

Let $u : \mathcal{X} \to \mathbb{R}$ be a scalar-valued function defined on an open set $\mathcal{X} \subseteq \mathbb{R}^d$ with $d \gg 1$. Our goal is to construct a feature map $g : \mathcal{X} \to \mathbb{R}^m$ with $m \ll d$ such that, given a prescribed tolerance $\varepsilon > 0$, there exists a function $f : \mathbb{R}^m \to \mathbb{R}$ for which

$$\mathbb{E}\left[(u(\boldsymbol{X}) - f(g(\boldsymbol{X})))^2\right] \leq \varepsilon^2. \tag{1}$$

Here, $\boldsymbol{X}$ denotes a random vector with probability density function π such that $\mathrm{supp}(\pi) = \mathcal{X}$, and $\mathbb{E}[\cdot]$ denotes the mathematical expectation. The function f is called the profile function and m the intermediate dimension. The construction of the profile function is postponed to Section 3.3, and we focus here on how to find a suitable feature map g such that (1) is attainable for some f. We note that the f which minimizes the above mean squared error is the conditional expectation $f : \boldsymbol{z} \mapsto \mathbb{E}[u(\boldsymbol{X})|g(\boldsymbol{X}) = \boldsymbol{z}]$. This well-known result will be used later. We now give two trivial solutions to (1) which help to understand the problem:

- With $g = \mathrm{Id}$, the identity function on $\mathcal{X}$, the profile function $f = u$ yields $f \circ g = u$. In this case we have $m = d$.
- With $g = u$, the profile $f = \mathrm{Id}$ also yields $f \circ g = u$ with an intermediate dimension $m = 1$.

Those two trivial solutions are not satisfactory either because $m = d \gg 1$ is large or because the computation of $g = u$ is untractable. The balance between the intermediate dimension m and the complexity of the feature map g appears as a central question in dimension reduction. Our goal is to construct g in a tractable space $\mathcal{G}_m$ of functions from $\mathcal{X}$ to $\mathbb{R}^m$. For instance, $\mathcal{G}_m$ could be a space of multivariate polynomial functions, a reproducing kernel Hilbert space, etc. We emphasize the necessity of *constraining* the function g to belong to a space of tractable functions; otherwise problem (1) makes no sense, as it admits a trivial solution with $g = u$.

2.2 Aligned gradients

From now on, we assume that $u : \mathcal{X} \to \mathbb{R}$ is continuously differentiable over the open set $\mathcal{X} \subseteq \mathbb{R}^d$ and that all the functions in $\mathcal{G}_m$ are also continuously differentiable.

Assumption 2.1. $u \in C^1(\mathcal{X}; \mathbb{R})$ and $g \in \mathcal{G}_m \subseteq C^1(\mathcal{X}; \mathbb{R}^m)$.

Let us assume for a moment that u is exactly of the form $u = f \circ g$ for some $g : \mathcal{X} \to \mathbb{R}^m$ and $f : \mathbb{R}^m \to \mathbb{R}$. Denoting by $\nabla f(\boldsymbol{z}) \in \mathbb{R}^m$ the gradient of f at point $\boldsymbol{z} \in \mathbb{R}^m$, and by

$$\nabla g(\boldsymbol{x}) = \begin{pmatrix} \nabla g_1(\boldsymbol{x})^T \\ \vdots \\ \nabla g_m(\boldsymbol{x})^T \end{pmatrix} \in \mathbb{R}^{m \times d},$$

the Jacobian[1] of g at point $\boldsymbol{x} \in \mathcal{X}$, the chain rule allows writing $\nabla u(\boldsymbol{x}) = \nabla g(\boldsymbol{x})^T \nabla f(g(\boldsymbol{x}))$ for any $\boldsymbol{x} \in \mathcal{X}$. In this case, $\nabla u(\boldsymbol{x})$ lies in the subspace $\mathrm{range}(\nabla g(\boldsymbol{x})^T)$ for any $\boldsymbol{x} \in \mathcal{X}$. In short, we have

$$u = f \circ g \quad \Longrightarrow \quad \nabla u(\boldsymbol{x}) \in \mathrm{range}(\nabla g(\boldsymbol{x})^T), \quad \forall \boldsymbol{x} \in \mathcal{X}.$$

[1] We use the standard convention that each row of the Jacobian matrix is the transpose of the gradient of each component.

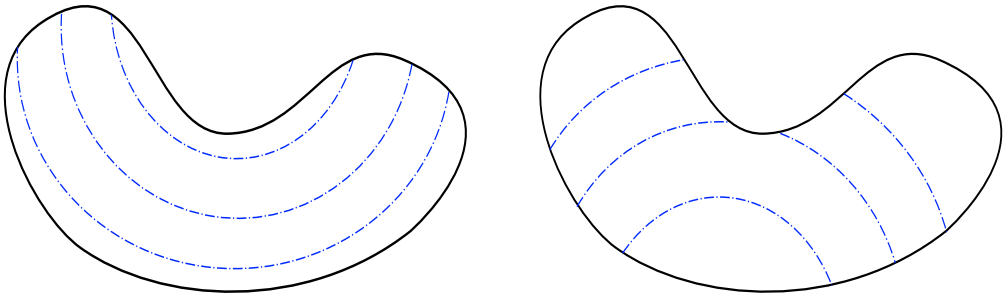


Figure 1: Illustration of Assumption 2.2: the black line represents the parameter space $\mathcal{X}$ and the blue lines represent different pre-images of two candidate functions g. On the left, the function g satisfies Assumption 2.2, but on the right, the level sets of the function g are not pathwise-connected.

Conversely, one can ask whether a function u which satisfies $\nabla u(\boldsymbol{x}) \in \text{range}(\nabla g(\boldsymbol{x})^T)$ for some vector-valued differentiable function g is necessarily of the form of $u = f \circ g$ for some f. The following proposition gives a positive answer to this question, under additional assumptions on g.

Assumption 2.2. The pre-image under g of any point is *smoothly pathwise-connected*; that is, for any $\boldsymbol{z} \in \text{Im}(g) \subseteq \mathbb{R}^m$ and for any points $\boldsymbol{x}, \boldsymbol{y}$ in the preimage $g^{-1}(\boldsymbol{z}) = \{\boldsymbol{s} \in \mathcal{X} : g(\boldsymbol{s}) = \boldsymbol{z}\}$, there exists a continuously differentiable function $\gamma : [0,1] \to g^{-1}(\boldsymbol{z})$ such that $\gamma(0) = \boldsymbol{x}$ and $\gamma(1) = \boldsymbol{y}$.

Proposition 2.3. Under Assumptions 2.1 and 2.2, if $u : \mathcal{X} \to \mathbb{R}$ and $g : \mathcal{X} \to \mathbb{R}^m$ satisfy

$$\nabla u(\boldsymbol{x}) \in \text{range}(\nabla g(\boldsymbol{x})^T), \tag{2}$$

for any $\boldsymbol{x} \in \mathcal{X}$, then $u = f \circ g$ for some function $f : \mathbb{R}^m \to \mathbb{R}$.

Proof. We first show that relation (2) implies the following property: if $g(\boldsymbol{x}) = g(\boldsymbol{y})$ for some $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$, then $u(\boldsymbol{x}) = u(\boldsymbol{y})$. Thus, let $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ be any two points such that $g(\boldsymbol{x}) = g(\boldsymbol{y})$. By Assumption 2.2, the pre-image $g^{-1}(\boldsymbol{z}), \boldsymbol{z} = g(\boldsymbol{x})$, is smoothly pathwise-connected so that there exsits a continuously differentiable path $\gamma : [0,1] \to \mathcal{X}$ from $\boldsymbol{x} = \gamma(0)$ to $\boldsymbol{y} = \gamma(1)$ such that $g(\gamma(t)) = \boldsymbol{z}$ for any $t \in [0,1]$. For any $1 \leq i \leq m$ the function $g_i \circ \gamma : [0,1] \to \mathbb{R}$ is constant so that $(g_i \circ \gamma)'(t) = \nabla g_i(\gamma(t))^T \gamma'(t) = 0$ for any $t \in [0,1]$, where $\gamma'(t) \in \mathbb{R}^d$ denotes the derivative of γ at point t. This means that, for any $t \in [0,1]$, the vector $\gamma'(t)$ is orthogonal to $\text{span}\{\nabla g_1(\gamma(t)), \ldots, \nabla g_m(\gamma(t))\} = \text{range}(\nabla g(\gamma(t))^T)$. By (2) we then have

$$(u \circ \gamma)'(t) = \nabla u(\gamma(t))^T \gamma'(t) = 0,$$

which implies that the continuous function $u \circ \gamma : [0,1] \to \mathbb{R}$ is constant. Then $u(\boldsymbol{x}) = u(\gamma(0)) = u(\gamma(1)) = u(\boldsymbol{y})$.

Now we build a function $f : \mathbb{R}^m \to \mathbb{R}$ such that $u = f \circ g$. Such a function needs to be defined only on the image $g(\mathcal{X}) \subseteq \mathbb{R}^m$ and can be set to zero on the complement of $g(\mathcal{X})$ in $\mathbb{R}^m$. We define f such that for any $\boldsymbol{z} \in g(\mathcal{X})$, $f(\boldsymbol{z}) = u(\boldsymbol{x})$ where $\boldsymbol{x} \in \mathcal{X}$ is any point such that $g(\boldsymbol{x}) = \boldsymbol{z}$. Even if this $\boldsymbol{x}$ is not unique, $f(\boldsymbol{z})$ is uniquely defined because $u(\boldsymbol{x}) = u(\boldsymbol{y})$ whenever $g(\boldsymbol{y}) = g(\boldsymbol{x})$. By construction we have $f(g(\boldsymbol{x})) = u(\boldsymbol{x})$ for any $\boldsymbol{x} \in \mathcal{X}$, which concludes the proof. □

Let us note that Assumption 2.2 is a necessary condition in Proposition 2.3. Indeed, if the pre-images of g are not smoothly pathwise-connected, as in the right plot of Figure 1, one can build

a function u which satisfies (2) without being of the form $f \circ g$. For example, think of a smooth function u which is constant on each of the connected parts of $g^{-1}(\boldsymbol{z})$ (so that (2) is satisfied) but which takes different values on each of those connected parts (so that $u \neq f \circ g$).

Here are some examples where Assumption (2.2) is satisfied.

Example 2.4 (Affine feature map)**.** Any function $g(\boldsymbol{x}) = A\boldsymbol{x} + b$ with $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$ satisfies Assumption 2.2, provided $\mathcal{X}$ is a convex set. Indeed, for any $\boldsymbol{z} \in \mathbb{R}^m$, $\boldsymbol{x}, \boldsymbol{y} \in g^{-1}(\boldsymbol{z})$, and $t \in [0,1]$, the quantity $\gamma(t) \coloneqq t\boldsymbol{x} + (1-t)\boldsymbol{y}$ belongs to $\mathcal{X}$ and it satisfies $g(\gamma(t)) = t(A\boldsymbol{x}+b) + (1-t)(A\boldsymbol{y}+b) = \boldsymbol{z}$, which shows that γ is a continuously differentiable path in $g^{-1}(\boldsymbol{z})$ from $\boldsymbol{x}$ to $\boldsymbol{y}$.

Example 2.5 (Feature map following from a C^1-diffeomorphism)**.** Assume $\mathcal{X}$ is convex. One way to build functions which satisfy Assumption 2.2 is to consider a C^1-diffeomorphism $\phi : \mathcal{X} \to \mathcal{X}$, meaning a continuously differentiable invertible function whose inverse is continuously differentiable, and to define $g(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \dots, \phi_m(\boldsymbol{x}))$ where $\phi_i(\boldsymbol{x})$ is the i-th component of $\phi(\boldsymbol{x})$. Such a g satisfies Assumption 2.2: for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ such that $g(\boldsymbol{x}) = g(\boldsymbol{y}) = \boldsymbol{z}$, the function

$$\gamma(t) = \phi^{-1}\big(t\phi(\boldsymbol{y}) + (1-t)\phi(\boldsymbol{x})\big),$$

defined for $t \in [0,1]$ is a smooth path from $\boldsymbol{x} = \gamma(0)$ to $\boldsymbol{y} = \gamma(1)$ as a composition of smooth functions. It is well defined because $t\phi(\boldsymbol{y}) + (1-t)\phi(\boldsymbol{x})$ is in $\mathcal{X}$ by convexity. By construction we have $\phi(\gamma(t)) = t\phi(\boldsymbol{y}) + (1-t)\phi(\boldsymbol{x})$ and the m first components of that relation yield $g(\gamma(t)) = tg(\boldsymbol{y}) + (1-t)g(\boldsymbol{x}) = \boldsymbol{z}$. This shows that $\gamma(t) \in g^{-1}(\boldsymbol{z})$, so that g satisfies Assumption 2.2.

Example 2.6 (Polynomial feature map)**.** Consider the case where g is a polynomial function on $\mathcal{X} = \mathbb{R}^d$. Assumption 2.2 is satisfied if and only if for any $\boldsymbol{z} \in g(\mathcal{X})$, the zeros of the polynomial $x \mapsto g(x) - z$ are pathwise-connected. Calculating the number of connected components (i.e., the zeroth Betti number) of an algebraic set like $\{\boldsymbol{x} : g(\boldsymbol{x}) - \boldsymbol{z} = 0\}$ is a difficult question, commonly encountered in algebraic geometry. Unfortunately, there is no easy answer to this question; see [37]. Still, we show later in Section 4 that polynomials work well from a numerical point of view, even though Assumption 2.2 is not checked in practice.

2.3 Aligning the gradients

Motivated by Proposition 2.3, we propose to build g by minimizing a cost function which measures how "aligned" are the gradient $\nabla u(\boldsymbol{x})$ and the subspace $\mathrm{range}(\nabla g(\boldsymbol{x})^T)$. For any $g : \mathcal{X} \to \mathbb{R}^m$ we introduce the cost function

$$J(g) = \mathbb{E}\left[\left\|\nabla u(\boldsymbol{X}) - \Pi_{\mathrm{range}(\nabla g(\boldsymbol{X})^T)} \nabla u(\boldsymbol{X})\right\|^2\right], \tag{3}$$

where $\Pi_{\mathrm{range}(\nabla g(\boldsymbol{X})^T)} \in \mathbb{R}^{d \times d}$ denotes the orthogonal projector onto $\mathrm{range}(\nabla g(\boldsymbol{X})^T)$ and $\|\cdot\|$ is the Euclidean norm on $\mathbb{R}^d$. Obviously we have $J(g) \geq 0$. The following proposition shows that if $J(g) = 0$ then there exists a profile function f such that $u = f \circ g$.

Proposition 2.7. Let $u : \mathcal{X} \to \mathbb{R}$ and $g : \mathcal{X} \to \mathbb{R}^m$ be continuously differentiable functions such that $J(g) = 0$. If g satisfies Assumption 2.2 and if

$$\mathrm{rank}(\nabla g(\boldsymbol{x})^T) = m, \tag{4}$$

for any $\boldsymbol{x} \in \mathcal{X}$, then there exists a function $f : \mathbb{R}^m \to \mathbb{R}$ such that $u = f \circ g$.

Before we give the proof of Proposition 2.7, let us comment on condition (4). This condition is commonly encountered in implicit function theory. It ensures that, for all $\boldsymbol{z} \in g(\mathcal{X})$, the level set $g^{-1}(\boldsymbol{z})$ is a smooth manifold of dimension $d-m$; see for instance Theorem 4.3.1 in [22]. One can easily check that (4) is satisfied in the case of affine feature maps $g(\boldsymbol{x}) = A\boldsymbol{x}+b$ with $\operatorname{rank}(A) = m$, but also in the case of feature maps following from a C^1-diffeomorphism; see Example 2.5.

Proof of Proposition 2.7. Let us assume for a moment that $\boldsymbol{x} \mapsto \Pi_{\operatorname{range}(\nabla g(\boldsymbol{x})^T)}$ is a continuous function from $\mathcal{X}$ to $\mathbb{R}^{d\times d}$. Then $\boldsymbol{x} \mapsto \|\nabla u(\boldsymbol{x}) - \Pi_{\operatorname{range}(\nabla g(\boldsymbol{x})^T)}\nabla u(\boldsymbol{x})\|$ is a continuous function, via products and sums of continuous functions. As $J(g) = 0$, then $\left\|\nabla u(\boldsymbol{x}) - \Pi_{\operatorname{range}(\nabla g(\boldsymbol{x})^T)}\nabla u(\boldsymbol{x})\right\|$ is equal to zero π-almost surely. By continuity, we have that $\|\nabla u(\boldsymbol{x}) - \Pi_{\operatorname{range}(\nabla g(\boldsymbol{x})^T)}\nabla u(\boldsymbol{x})\|$ is equal to zero for all $\boldsymbol{x} \in \operatorname{supp}(\pi) = \mathcal{X}$, so that $\nabla u(\boldsymbol{x}) \in \operatorname{range}(\nabla g(\boldsymbol{x})^T)$ holds for any $\boldsymbol{x} \in \mathcal{X}$. Together with Assumption 2.2, Proposition 2.3 ensures the existence of $f : \mathbb{R}^m \to \mathbb{R}$ such that $u = f \circ g$.

It remains to show that $\boldsymbol{x} \mapsto \Pi_{\operatorname{range}(\nabla g(\boldsymbol{x})^T)}$ is continuous. Let $M(\boldsymbol{x}) = \nabla g(\boldsymbol{x})\nabla g(\boldsymbol{x})^T \in \mathbb{R}^{m\times m}$. By Assumption (4) $M(\boldsymbol{x})$ is invertible and we can write $\Pi_{\operatorname{range}(\nabla g(\boldsymbol{x})^T)} = \nabla g(\boldsymbol{x})^T M(\boldsymbol{x})^{-1}\nabla g(\boldsymbol{x})$ for any $\boldsymbol{x} \in \mathcal{X}$. For any $\delta \in \mathbb{R}^d$ we can write

$$\begin{aligned}\|M(\boldsymbol{x})^{-1} - M(\boldsymbol{x}+\delta)^{-1}\|_{\mathrm{sp}} &\leq \|M(\boldsymbol{x}+\delta)^{-1}\|_{\mathrm{sp}}\|M(\boldsymbol{x}+\delta)M(\boldsymbol{x})^{-1} - I_d\|_{\mathrm{sp}} \\ &= \lambda_{\min}(M(\boldsymbol{x}+\delta))^{-1}\|M(\boldsymbol{x}+\delta)M(\boldsymbol{x})^{-1} - I_d\|_{\mathrm{sp}},\end{aligned}$$

where $\|\cdot\|_{\mathrm{sp}}$ denotes the spectral norm and where $\lambda_{\min}(M(\boldsymbol{x}+\delta))$ denotes the smallest eigenvalue of $M(\boldsymbol{x}+\delta)$. Because the eigenvalues are continuous with respect to the matrix entries (see [38]) and by Assumption (4), we have $\lambda_{\min}(M(\boldsymbol{x}+\delta)) \to \lambda_{\min}(M(\boldsymbol{x})) > 0$ as $\delta \to 0$. Therefore we have $\lambda_{\min}(M(\boldsymbol{x}+\delta))^{-1}\|M(\boldsymbol{x}+\delta)M(\boldsymbol{x})^{-1} - I_d\|_{\mathrm{sp}} \to \lambda_{\min}(M(\boldsymbol{x}))^{-1}\|I_d - I_d\|_{\mathrm{sp}} = 0$. This shows the continuity of $\boldsymbol{x} \mapsto M(\boldsymbol{x})^{-1}$ and therefore the continuity of $\boldsymbol{x} \mapsto \Pi_{\operatorname{range}(\nabla g(\boldsymbol{x})^T)} = \nabla g(\boldsymbol{x})^T M(\boldsymbol{x})^{-1}\nabla g(\boldsymbol{x})$. This concludes the proof. □

Next we consider the minimization problem

$$\min_{g\in\mathcal{G}_m} J(g), \tag{5}$$

where $\mathcal{G}_m \subseteq C^1(\mathcal{X};\mathbb{R}^m)$ is a set of tractable functions. In general, given some choice of $\mathcal{G}_m$, the minimum of the cost function will not be exactly zero, and thus an assumption of Proposition 2.7 will not hold. Using arguments based on Poincaré inequalities, Proposition 2.9 below shows that, under specific assumptions, there exists at least one function $f : \mathbb{R}^m \to \mathbb{R}$ such that $\mathbb{E}[(u(\boldsymbol{X}) - f(g(\boldsymbol{X})))^2]$ is of the same order of magnitude as $J(g)$. In other words, we will be able to control the L^2-error in an approximation of u by making $J(g)$ small. Let us first introduce the Poincaré inequality associated with a random variable.

Definition 2.8 (Poincaré inequality)**.** Given a continuous random variable $\boldsymbol{X}_{\mathcal{M}}$ taking values in a smooth manifold $\mathcal{M}$, the *Poincaré constant* $\mathbb{C}(\boldsymbol{X}_{\mathcal{M}}) \in [0,+\infty]$ is defined as the smallest constant such that

$$\mathbb{E}\left[\left(h(\boldsymbol{X}_{\mathcal{M}}) - \mathbb{E}[h(\boldsymbol{X}_{\mathcal{M}})]\right)^2\right] \leq \mathbb{C}(\boldsymbol{X}_{\mathcal{M}})\,\mathbb{E}\left[\left\|\nabla h(\boldsymbol{X}_{\mathcal{M}})\right\|^2\right] \tag{6}$$

holds for any continuously differentiable function $h : \mathcal{M} \to \mathbb{R}$. Here, the gradient $\nabla h(\boldsymbol{z})$ is a vector in $T_{\boldsymbol{z}}(\mathcal{M})$, the tangent space of $\mathcal{M}$ at point $\boldsymbol{z} \in \mathcal{M}$. We say that $\boldsymbol{X}_{\mathcal{M}}$ satisfies the *Poincaré inequality* (6) if $\mathbb{C}(\boldsymbol{X}_{\mathcal{M}}) < +\infty$.

We refer to [4] for a simple proof of the Poincaré inequality for a large class of probability measures.

Proposition 2.9. Assume that the set of functions $\mathcal{G}_m \subseteq C^1(\mathcal{X};\mathbb{R}^m)$ is such that $\operatorname{rank}(\nabla g(\boldsymbol{x})^T) = m$ for all $g \in \mathcal{G}_m$ and all $\boldsymbol{x} \in \mathcal{X}$. Furthermore, assume that $\mathcal{G}_m$ satisfies

$$\mathbb{C}(\boldsymbol{X}|\mathcal{G}_m) \coloneqq \sup_{g\in\mathcal{G}_m} \sup_{\boldsymbol{z}\in g(\mathcal{X})} \mathbb{C}(\boldsymbol{X} \,|\, g(\boldsymbol{X}) = \boldsymbol{z}) \; < \infty, \tag{7}$$

where $\boldsymbol{X} \,|\, g(\boldsymbol{X}) = \boldsymbol{z}$ denotes the random variable obtained by conditioning $\boldsymbol{X}$ on the event $g(\boldsymbol{X}) = \boldsymbol{z}$. Then, for any $g \in \mathcal{G}_m$, there exists a measurable $f : \mathbb{R}^m \to \mathbb{R}$ such that

$$\mathbb{E}\left[\big(u(\boldsymbol{X}) - f(g(\boldsymbol{X}))\big)^2\right] \leq \mathbb{C}(\boldsymbol{X}|\mathcal{G}_m) J(g), \tag{8}$$

where $J(g)$ is defined as in (3).

Proof of Proposition 2.9. Let $g \in \mathcal{G}_m$. Because $\operatorname{rank}(\nabla g(\boldsymbol{x})^T) = m$ for any $\boldsymbol{x} \in \mathcal{X}$, the level set $\mathcal{M} = g^{-1}(\boldsymbol{z})$ for some $\boldsymbol{z} \in g(\mathcal{X})$ is a smooth manifold of dimension $d - m$; see Theorem 4.3.1 in [22]. Let $u_{\mathcal{M}} : \mathcal{M} \to \mathbb{R}$ be the restriction of u to $\mathcal{M}$. Together with (7), the Poincaré inequality (6) with $h = u_{\mathcal{M}}$ and $\boldsymbol{X}_{\mathcal{M}} = (\boldsymbol{X}|g(\boldsymbol{X}) = \boldsymbol{z})$ permits writing

$$\begin{aligned}\mathbb{E}[(u(\boldsymbol{X}_{\mathcal{M}}) - \mathbb{E}[u(\boldsymbol{X}_{\mathcal{M}})])^2] &= \mathbb{E}[(u_{\mathcal{M}}(\boldsymbol{X}_{\mathcal{M}}) - \mathbb{E}[u_{\mathcal{M}}(\boldsymbol{X}_{\mathcal{M}})])^2] \\ &\overset{(6)\&(7)}{\leq} \mathbb{C}(\boldsymbol{X}|\mathcal{G}_m)\, \mathbb{E}[\|\nabla u_{\mathcal{M}}(\boldsymbol{X}_{\mathcal{M}})\|^2]. \end{aligned} \tag{9}$$

Because $\mathcal{M}$ is a smooth manifold embedded in $\mathbb{R}^d$, the gradient $\nabla u_{\mathcal{M}}$ can be expressed by means of the gradient ∇u as follows

$$\nabla u_{\mathcal{M}}(\boldsymbol{x}) = \Pi_{T_{\boldsymbol{x}}(\mathcal{M})} \nabla u(\boldsymbol{x}) \tag{10}$$

for all $\boldsymbol{x} \in \mathcal{M}$, where $\Pi_{T_{\boldsymbol{x}}(\mathcal{M})} \in \mathbb{R}^{d\times d}$ is the orthogonal projector onto $T_{\boldsymbol{x}}(\mathcal{M})$, the tangent space of $\mathcal{M}$ at $\boldsymbol{x}$. Since $\mathcal{M}$ is a level set of g, we have $T_{\boldsymbol{x}}(\mathcal{M}) = \ker(\nabla g(\boldsymbol{x})) = (\operatorname{range}(\nabla g(\boldsymbol{x})^T))^\perp$ (see for instance [1, Section 3.5.7]) so that

$$\Pi_{T_{\boldsymbol{x}}(\mathcal{M})} = \Pi_{\ker(\nabla g(\boldsymbol{x}))} = I_d - \Pi_{\operatorname{range}(\nabla g(\boldsymbol{x})^T)}. \tag{11}$$

Combining (9) with (10) and (11) we obtain

$$\mathbb{E}[(u(\boldsymbol{X}_{\mathcal{M}}) - \mathbb{E}[u(\boldsymbol{X}_{\mathcal{M}})])^2] \leq \mathbb{C}(\boldsymbol{X}|\mathcal{G}_m)\; \mathbb{E}[\|(I_d - \Pi_{\operatorname{range}(\nabla g(\boldsymbol{X}_{\mathcal{M}})^T)})\nabla u(\boldsymbol{X}_{\mathcal{M}})\|^2]. \tag{12}$$

Now, because $\boldsymbol{X}_{\mathcal{M}}$ is the conditional random variable $\boldsymbol{X}|g(\boldsymbol{X}) = \boldsymbol{z}$, we can interpret any expectation $\mathbb{E}[\phi(\boldsymbol{X}_{\mathcal{M}})]$ as a conditional expectation $\mathbb{E}[\phi(\boldsymbol{X})|g(\boldsymbol{X}) = \boldsymbol{z}]$ for any integrable function $\phi : \mathcal{X} \to \mathbb{R}$. This manipulation permits rewriting the inequality (12) as

$$\begin{aligned}&\mathbb{E}[(u(\boldsymbol{X}) - \mathbb{E}[u(\boldsymbol{X})|g(\boldsymbol{X})])^2 \,|\, g(\boldsymbol{X}) = \boldsymbol{z}] \\ &\qquad \leq \mathbb{C}(\boldsymbol{X}|\mathcal{G}_m)\; \mathbb{E}\left[\|(I_d - \Pi_{\operatorname{range}(\nabla g(\boldsymbol{X})^T)})\nabla u(\boldsymbol{X})\|^2 \,\middle|\, g(\boldsymbol{X}) = \boldsymbol{z}\right]\end{aligned}$$

Replacing $\boldsymbol{z}$ by the random variable $\boldsymbol{Z} = g(\boldsymbol{X})$ and taking the expectation on both sides, we obtain

$$\mathbb{E}\left[(u(\boldsymbol{X}) - \mathbb{E}[u(\boldsymbol{X})|g(\boldsymbol{X})])^2\right] \leq \mathbb{C}(\boldsymbol{X}|\mathcal{G}_m)\; \mathbb{E}\left[\|(I_d - \Pi_{\operatorname{range}(\nabla g(\boldsymbol{X})^T)})\nabla u(\boldsymbol{X})\|^2\right].$$

Finally we define the measurable function $f : \mathbb{R}^m \to \mathbb{R}$ such that $f(\boldsymbol{z}) = \mathbb{E}[u(\boldsymbol{X})|g(\boldsymbol{X}) = \boldsymbol{z}]$ for any $\boldsymbol{z} \in \mathbb{R}^m$. We can write $\mathbb{E}[u(\boldsymbol{X})|g(\boldsymbol{X})] = f(g(\boldsymbol{X}))$ which yields (8) and concludes the proof. □

Proposition 2.9 ensures that, for any $g \in \mathcal{G}_m$, there exists a function $f : \mathbb{R}^m \to \mathbb{R}$ such that the mean squared error between u and $f \circ g$ is bounded by $\mathbb{C}(\boldsymbol{X}|\mathcal{G}_m)J(g)$. This remarkable property justifies the use of the cost function J for the construction of g.

Remark 2.10 (Linear feature maps and the Gaussian distribution). When $X \sim \mathcal{N}(0, I_d)$ is a standard Gaussian random vector and when $\mathcal{G}_m = \{\boldsymbol{x} \mapsto U\boldsymbol{x} : U \in \mathbb{R}^{m\times d}, UU^T = I_m\}$ contains linear features, the constant $\mathbb{C}(\boldsymbol{X}|\mathcal{G}_m)$ is equal to 1. Indeed, the level sets $g^{-1}(\boldsymbol{z})$ are affine subspaces and any conditional random variable of the form $\boldsymbol{X}|g(\boldsymbol{X}) = \boldsymbol{z}$ is Gaussian with identity covariance. Theorem 3.20 in [6] ensures that $\mathbb{C}(\boldsymbol{X}|g(\boldsymbol{X}) = \boldsymbol{z}) = 1$ for any $g \in \mathcal{G}_m$ and $\boldsymbol{z} \in g(\mathcal{X})$, which yields $\mathbb{C}(\boldsymbol{X}|\mathcal{G}_m) = 1$.

We conclude this section with an important property of J. Consider a $\mathcal{C}^1$-diffeomorphism $\phi : \mathbb{R}^m \to \mathbb{R}^m$. Since $\nabla\phi(\boldsymbol{x}) \in \mathbb{R}^{m\times m}$ is invertible for all $\boldsymbol{x} \in \mathcal{X}$, it holds that $\text{range}(\nabla\phi \circ g(\boldsymbol{X})^T) = \text{range}(\nabla g(\boldsymbol{X})^T \nabla\phi(g(\boldsymbol{X}))^T) = \text{range}(\nabla g(\boldsymbol{X})^T)$. Thus we have

$$J(\phi \circ g) = J(g). \tag{13}$$

This invariance reflects the following property of our initial dimension reduction problem (1): any composed function $f \circ g$ can be written as the composition of $f \circ \phi^{-1}$ with $\phi \circ g$ so that the feature maps g and $\phi \circ g$ are equivalent with regard to the problem (1). The invariance (13) offers the possibility to arbitrarily impose the probability law of $g(\boldsymbol{X})$. Indeed, under natural assumptions on g, there exists a $\mathcal{C}^1$-diffeomorphism $\phi = \phi_g$ depending on g so that $\phi_g \circ g(\boldsymbol{X})$ follows, for instance, the standard normal distribution $\mathcal{N}(0, I_d)$; see [41]. Replacing g by $\bar{g} = \phi_g \circ g$ yields the same value of $J(\bar{g}) = J(g)$ with $\bar{g}(\boldsymbol{X}) \sim \mathcal{N}(0, I_d)$. However, constructing ϕ_g can be numerically expensive in practice. A more pragmatic way to exploit (13) is simply to consider the affine transformation $\phi_g(\boldsymbol{z}) = \text{Cov}(g(\boldsymbol{X}))^{-1/2}(\boldsymbol{z} - \mathbb{E}[g(\boldsymbol{X})])$, which ensures that $\phi_g \circ g(\boldsymbol{X})$ is centered with identity covariance. This affine map is readily computable and allows one to normalize the feature map g. In the following, we will consider the constrained minimization problem

$$\min_{\substack{g\in\mathcal{G}_m \\ \mathbb{E}[g(\boldsymbol{X})]=0 \\ \text{Cov}(g(\boldsymbol{X}))=I_d}} J(g). \tag{14}$$

The constraints $\mathbb{E}[g(\boldsymbol{X})] = 0$ and $\text{Cov}(g(\boldsymbol{X})) = I_d$ will be useful to stabilize the minimization algorithms, as described in the next section.

3 Algorithms

Based on the previous section, an approximation $f \circ g$ of u can be obtained by first minimizing $J(g)$ over some prescribed feature map space $\mathcal{G}_m$, and then by minimizing the mean squared error $\mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2]$ over $f \in \mathcal{F}_m$. In this section we propose adaptive algorithms to construct a feature map space $\mathcal{G}_m$ of the form

$$\mathcal{G}_m = \left\{ g : \boldsymbol{x} \mapsto \begin{pmatrix} g_1(\boldsymbol{x}) \\ \vdots \\ g_m(\boldsymbol{x}) \end{pmatrix} \text{ where } g_i \in \text{span}\{\Phi_1, \dots, \Phi_K\} \right\} \tag{15}$$

and a profile function space $\mathcal{F}_m$ of the form

$$\mathcal{F}_m = \text{span}\{\Psi_1, \dots, \Psi_P\}, \tag{16}$$

where $\Phi_1, \dots, \Phi_K$ and $\Psi_1, \dots, \Psi_P$ are polynomials defined on $\mathbb{R}^d$ and $\mathbb{R}^m$, respectively. In practice we make use of a sample $\{(\boldsymbol{x}^{(i)}, u(\boldsymbol{x}^{(i)}), \nabla u(\boldsymbol{x}^{(i)})\}_{i=1}^N$ of size N, which allows estimating $J(g)$ by

$$\widehat{J}(g) \coloneqq \frac{1}{N} \sum_{i=1}^{N} \|\nabla u(\boldsymbol{x}^{(i)}) - \Pi_{\mathrm{range}(\nabla g(\boldsymbol{x}^{(i)})^T)} \nabla u(\boldsymbol{x}^{(i)})\|^2, \tag{17}$$

and the mean squared error $\mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2]$ by $\frac{1}{N}\sum_{i=1}^N (u(\boldsymbol{x}^{(i)}) - f \circ g(\boldsymbol{x}^{(i)}))^2$. First we present in Section 3.1 an algorithm for the minimization of $\widehat{J}(g)$ over a given (fixed) space $\mathcal{G}_m$. Then in Section 3.2 we propose a greedy procedure to enrich the space $\mathcal{G}_m$ adaptively. A similar procedure will be presented in Section 3.3 for the construction of the polynomial space $\mathcal{F}_m$. For those adaptive algorithms, a cross-validation error analysis determines when to stop the enrichment procedures, as described in Section 3.4.

3.1 Maximizing the expectation of a Rayleigh quotient

Assume the basis $\{\Phi_1, \dots, \Phi_K\}$ of the feature map space (15) is given, with $K \geq m$. We show that minimizing $J(g)$ (or $\widehat{J}(g)$) over $g \in \mathcal{G}_m$ boils down to the maximization of the expectation of a generalized Rayleigh quotient. We then propose a quasi-Newton algorithm to solve the problem.

With the notation $\Phi(\boldsymbol{x}) = (\Phi_1(\boldsymbol{x}), \dots, \Phi_K(\boldsymbol{x})) \in \mathbb{R}^K$, any feature map g in the space $\mathcal{G}_m$ defined by (15) can be written as

$$g(\boldsymbol{x}) = G^T \Phi(\boldsymbol{x}),$$

for some matrix $G \in \mathbb{R}^{K \times m}$. In order to account for the constraints $\mathbb{E}[g(\boldsymbol{X})] = 0$ and $\mathrm{Cov}(g(\boldsymbol{X})) = I_d$ in (14), we assume that $\mathbb{E}[\Phi(\boldsymbol{X})] = 0$ and we impose the constraint that G satisfy

$$G^T \mathrm{Cov}(\Phi(\boldsymbol{X}))G = I_d. \tag{18}$$

Assuming the Jacobian $\nabla g(\boldsymbol{X}) = G^T \nabla \Phi(\boldsymbol{X})$ has rank m almost surely, the orthogonal projector $\Pi_{\mathrm{range}(\nabla g(\boldsymbol{X})^T)}$ can be expressed as

$$\Pi_{\mathrm{range}(\nabla g(\boldsymbol{X})^T)} = \nabla g(\boldsymbol{X})^T \left(\nabla g(\boldsymbol{X}) \nabla g(\boldsymbol{X})^T\right)^{-1} \nabla g(\boldsymbol{X}),$$

and the cost function $J(g)$ becomes

$$\begin{aligned}
J(g) &= \mathbb{E}\left[\left\|\nabla u(\boldsymbol{X}) - \Pi_{\mathrm{range}(\nabla g(\boldsymbol{X})^T)} \nabla u(\boldsymbol{X})\right\|^2\right] \\
&= \mathbb{E}\left[\|\nabla u(\boldsymbol{X})\|^2\right] - \mathbb{E}\left[\left\|\Pi_{\mathrm{range}(\nabla g(\boldsymbol{X})^T)} \nabla u(\boldsymbol{X})\right\|^2\right] \\
&= \mathbb{E}\left[\|\nabla u(\boldsymbol{X})\|^2\right] - \mathbb{E}\left[\nabla u(\boldsymbol{X})^T \nabla g(\boldsymbol{X})^T \left(\nabla g(\boldsymbol{X}) \nabla g(\boldsymbol{X})^T\right)^{-1} \nabla g(\boldsymbol{X}) \nabla u(\boldsymbol{X})\right] \\
&= \mathbb{E}\left[\|\nabla u(\boldsymbol{X})\|^2\right] - \mathbb{E}\left[\mathrm{trace}\left(G^T A(\boldsymbol{X}) G\right)\left(G^T B(\boldsymbol{X}) G\right)^{-1}\right].
\end{aligned}$$

Here, $A(\boldsymbol{X}) \in \mathbb{R}^{K \times K}$ and $B(\boldsymbol{X}) \in \mathbb{R}^{K \times K}$ are two symmetric positive semidefinite matrices given by

$$\begin{aligned}
A(\boldsymbol{X}) &= \nabla \Phi(\boldsymbol{X}) \nabla u(\boldsymbol{X}) \nabla u(\boldsymbol{X})^T \nabla \Phi(\boldsymbol{X})^T, \\
B(\boldsymbol{X}) &= \nabla \Phi(\boldsymbol{X}) \nabla \Phi(\boldsymbol{X})^T.
\end{aligned}$$

Minimizing $g \mapsto J(g)$ over $\mathcal{G}_m$ is the same as maximizing

$$\mathcal{R}(G) = \mathbb{E}\left[\mathrm{trace}\left(\left(G^T A(\boldsymbol{X}) G\right)\left(G^T B(\boldsymbol{X}) G\right)^{-1}\right)\right], \tag{19}$$

over $G \in \mathbb{R}^{K\times m}$. Similarily, minimizing $g \mapsto \widehat{J}(g)$ over $\mathcal{G}_m$ is the same as maximizing

$$\widehat{\mathcal{R}}(G) = \frac{1}{N}\sum_{i=1}^{N} \operatorname{trace}\left(\left(G^T A(\boldsymbol{X}^{(i)})G\right)\left(G^T B(\boldsymbol{X}^{(i)})G\right)^{-1}\right), \tag{20}$$

over $G \in \mathbb{R}^{K\times m}$. The quantity $\mathcal{R}(G)$ corresponds to the expectation of the generalized Rayleigh quotient associated with the matrix pair $(A(\boldsymbol{X}), B(\boldsymbol{X}))$, and $\widehat{\mathcal{R}}(G)$ to its Monte Carlo estimate. It is easier to recognize the generalized Rayleigh quotient when $m = 1$, since $G \in \mathbb{R}^K$ becomes a vector so that $\mathcal{R}(G) = \mathbb{E}[\frac{G^T A(\boldsymbol{X})G}{G^T B(\boldsymbol{X})G}]$ and $\widehat{\mathcal{R}}(G) = \frac{1}{N}\sum_{i=1}^{N} \frac{G^T A(\boldsymbol{x}^{(i)})G}{G^T B(\boldsymbol{x}^{(i)})G}$. Generalized Rayleigh quotients are ubiquitous in dimension reduction; see [21]. However, the *expectations* or *sums* of generalized Rayleigh quotients as in (19) and (20) are not common and appear to be much more difficult to maximize. As shown in [42, 48, 49], maximizing the sum of two generalized Rayleigh quotients is already a difficult task, which requires dedicated algorithms. In the particular case where the feature map is linear, however, maximizing $\mathcal{R}(G)$ can be done analytically, as shown by the next remark.

Remark 3.1 (Linear feature maps and active subspaces). The space of linear feature maps $\mathcal{G}_m = \{\boldsymbol{x} \mapsto G^T\boldsymbol{x} : G \in \mathbb{R}^{d\times m}\}$ corresponds to (15) with $\Phi(\boldsymbol{x}) = \boldsymbol{x}$, the identity map. In this case $\nabla\Phi(\boldsymbol{x}) = I_d$ is independent of $\boldsymbol{x}$ so that $A(\boldsymbol{X}) = \nabla u(\boldsymbol{X})\nabla u(\boldsymbol{X})^T$ and $B(\boldsymbol{X}) = I_d$. The expected generalized Rayleigh quotient (19) becomes the standard (matrix) Rayleigh quotient $\mathcal{R}(G) = \operatorname{trace}((G^T HG)(G^T G)^{-1})$ where

$$H = \mathbb{E}[\nabla u(\boldsymbol{X})\nabla u(\boldsymbol{X})^T].$$

The maximum of $G \mapsto \mathcal{R}(G)$ is known to be attained by any matrix $G \in \mathbb{R}^{K\times m}$ whose columns span the m-dimensional dominant eigenspace of H. This subspace is sometimes called the active subspace; see [11, 12, 45]. When considering the sample approximation $\widehat{\mathcal{R}}(G)$ in (20), the matrix H is simply replaced by its approximation $\widehat{H} = \frac{1}{N}\sum_{i=1}^{N} \nabla u(\boldsymbol{x}^{(i)})\nabla u(\boldsymbol{x}^{(i)})^T$. The accuracy of the active subspace recovery from $\widehat{H}$ depends on the sample size N, on the active subspace dimension m, and on the spectrum of H; see [23] for more details.

So far we have seen that, provided the basis $\Phi(\boldsymbol{x}) = (\Phi_1(\boldsymbol{x}), \dots, \Phi_K(\boldsymbol{x}))$ satisfies $\mathbb{E}[\Phi(\boldsymbol{X})] = 0$, the minimization problem (14) can be rewritten as

$$\min_{\substack{g\in\mathcal{G}_m \\ \mathbb{E}[g(\boldsymbol{X})]=0 \\ \operatorname{Cov}(g(\boldsymbol{X}))=I_d}} J(g) \qquad \overset{g(\boldsymbol{x})=G^T\Phi(\boldsymbol{x})}{\Longleftrightarrow} \qquad \max_{\substack{G\in\mathbb{R}^{K\times m} \\ G^T\operatorname{Cov}(\Phi(\boldsymbol{X}))G=I_d}} \mathcal{R}(G). \tag{21}$$

Next we propose a quasi-Newton method to solve this problem. The following proposition gives the expression for the gradient of $G \mapsto \mathcal{R}(G)$. The proof is given in Appendix B.

Proposition 3.2. Let $A(\boldsymbol{X}), B(\boldsymbol{X}) \in \mathbb{R}^{K\times K}$ be two random symmetric positive semidefinite matrices. Assume that for a given $G \in \mathbb{R}^{K\times m}$, there exists $\varepsilon > 0$ such that $(G+\delta G)^T B(\boldsymbol{X})(G+\delta G)$ is almost surely invertible for any $\|\delta G\| \le \varepsilon$. Then $\mathcal{R}(\cdot)$ defined by (19) is differentiable at G and its gradient $\nabla\mathcal{R}(G) \in \mathbb{R}^{K\times m}$ is such that $(\nabla\mathcal{R}(G))_{ij} = \frac{\partial\mathcal{R}(G)}{\partial G_{ij}}$ can be written as

$$\nabla\mathcal{R}(G) = 2\left(\left(H(G) - \Sigma(G)\right)G_{\mathrm{vec}}\right)_{\mathrm{mat}}, \tag{22}$$

where $H(G)$ and $\Sigma(G)$ are two symmetric positive semidefinite matrices in $\mathbb{R}^{(Km)\times(Km)}$ given by

$$H(G) = \mathbb{E}\left[\left((G^T B(\boldsymbol{X})G)^{-1}\right) \otimes A(\boldsymbol{X})\right] \tag{23}$$

$$\Sigma(G) = \mathbb{E}\left[\left((G^T B(\boldsymbol{X})G)^{-1}G^T A(\boldsymbol{X})G(G^T B(\boldsymbol{X})G)^{-1}\right) \otimes B(\boldsymbol{X})\right]. \tag{24}$$

Here, the notation $(\cdot)_{\mathrm{vec}}$ denotes the vectorization of a matrix, such that $G_{\mathrm{vec}} \in \mathbb{R}^{Km}$ is the vertical concatenation of the columns of $G \in \mathbb{R}^{K\times m}$. The matricization $(\cdot)_{\mathrm{mat}}$ is the reverse operation, such that $(G_{\mathrm{vec}})_{\mathrm{mat}} = G$. The notation $\otimes$ denotes the Kronecker product.

Starting at an initial guess $G^{(0)} \in \mathbb{R}^{K\times m}$, a quasi-Newton method for maximizing $G \mapsto \mathcal{R}(G)$ is an iterative procedure $G^{(k+1)} = G^{(k)} - (\mathcal{H}^{(k)})^{-1}\nabla\mathcal{R}(G^{(k)})$ where $\mathcal{H}^{(k)} : \mathbb{R}^{K\times m} \to \mathbb{R}^{K\times m}$ is an approximation to the Hessian of $\mathcal{R}(\cdot)$ at point $G^{(k)}$; see [15]. Because our goal is to maximize $\mathcal{R}(\cdot)$, the operator $\mathcal{H}^{(k)}$ should be chosen symmetric negative definite. We propose to use $\mathcal{H}^{(k)} = -2\Sigma(G^{(k)})$. This matrix naturally appears in the expression of the Hessian $\nabla^2\mathcal{R}(G^{(k)})$ when differentiating the relation (22). Assuming $\Sigma(G^{(k)})$ is invertible (we observe in practice that it is non-singular) the quasi-Newton iteration in vectorized form is

$$\begin{aligned} G^{(k+1)}_{\mathrm{vec}} &= G^{(k)}_{\mathrm{vec}} - \Big(\big(\mathcal{H}^{(k)}\big)^{-1}\nabla\mathcal{R}(G^{(k)})\Big)_{\mathrm{vec}} \\ &\overset{(22)}{=} G^{(k)}_{\mathrm{vec}} - \Big(-2\Sigma(G^{(k)})\Big)^{-1}\Big(2H(G^{(k)}) - 2\Sigma(G^{(k)})\Big)G^{(k)}_{\mathrm{vec}} \\ &= \Sigma(G^{(k)})^{-1}H(G^{(k)})G^{(k)}_{\mathrm{vec}}. \end{aligned} \tag{25}$$

To account for the constraint $G^T\mathrm{Cov}(\Phi(\boldsymbol{X}))G = I_m$ in (21), notice that, by the definition (19) of $\mathcal{R}(\cdot)$, we have $\mathcal{R}(GM) = \mathcal{R}(G)$ for any invertible matrix $M \in \mathbb{R}^{m\times m}$. By letting $M = (G^T\mathrm{Cov}(\Phi(\boldsymbol{X}))G)^{-1/2}$, the matrix $\widetilde{G} = GM$ satisfies the constraint $\widetilde{G}^T\mathrm{Cov}(\Phi(\boldsymbol{X}))\widetilde{G} = I_m$ and yields the same Rayleigh quotient $\mathcal{R}(\widetilde{G}) = \mathcal{R}(G)$. Following this reasoning, we modify the iterations (25) by adding a normalization step:

$$G^{(k+1/2)} = \Big(\Sigma(G^{(k)})^{-1}H(G^{(k)})G^{(k)}_{\mathrm{vec}}\Big)_{\mathrm{mat}}, \tag{26}$$

$$G^{(k+1)} = G^{(k+1/2)}\Big(G^{(k+1/2)T}\mathrm{Cov}(\Phi(\boldsymbol{X}))G^{(k+1/2)}\Big)^{-1/2}. \tag{27}$$

Interestingly, this quasi-Newton procedure is very similar to a power iteration for solving eigenvalue problems; see the next remark.

Remark 3.3 (Quasi-Newton method and power iteration). Let us continue Remark 3.1, where $\mathcal{G}_m$ is the space of linear feature maps. Recall that $\Phi(\boldsymbol{x}) = \boldsymbol{x}$, $A(\boldsymbol{X}) = \nabla u(\boldsymbol{X})\nabla u(\boldsymbol{X})^T$, $B(\boldsymbol{X}) = I_d$, and assume for simplicity that $\mathrm{Cov}(\Phi(\boldsymbol{X})) = I_d$. Given an iterate $G^{(k)}$ such that $G^{(k)}G^{(k)T} = I_d$, the matrices $H(G^{(k)})$ and $\Sigma(G^{(k)})$ introduced in (23) and (24) become $H(G^{(k)}) = I_d \otimes H$ and $\Sigma(G^{(k)}) = (G^{(k)T}HG^{(k)}) \otimes I_d$, where $H = \mathbb{E}[\nabla u(\boldsymbol{X})\nabla u(\boldsymbol{X})^T]$. Using the relation $((S_2 \otimes S_1)G_{\mathrm{vec}})_{\mathrm{mat}} = S_1GS_2$ for any symmetric matrices S_1, S_2, the quasi-Newton iteration (26) becomes

$$G^{(k+1/2)} = \left(\left(\left(G^{(k)T}HG^{(k)}\right)^{-1} \otimes H\right)G^{(k)}_{\mathrm{vec}}\right)_{\mathrm{mat}} = HG^k\left(G^{(k)T}HG^{(k)}\right)^{-1}. \tag{28}$$

Thus, the relation

$$\mathrm{range}(G^{(k+1)}) \overset{(27)}{=} \mathrm{range}(G^{(k+1/2)}) \overset{(28)}{=} \mathrm{range}(HG^{(k)}) = \mathrm{range}(H^{k+1}G^{(0)})$$

holds and shows that the quasi-Newton iteration (26) with the normalization step (27) is precisely a power iteration method which aims to compute the m-dimensional dominant eigenspace of the matrix H.

In practice, the quasi-Newton method (26) and (27) can be used to maximize $\widehat{\mathcal{R}}(G)$ (20) by replacing $H(G)$ and $\Sigma(G)$ with their sample approximations:

$$\widehat{H}(G) = \frac{1}{N}\sum_{i=1}^{N}\left((G^T B(\boldsymbol{x}^{(i)})G)^{-1}\right) \otimes A(\boldsymbol{x}^{(i)})$$

$$\widehat{\Sigma}(G) = \frac{1}{N}\sum_{i=1}^{N}\left((G^T B(\boldsymbol{x}^{(i)})G)^{-1}G^T A(\boldsymbol{x}^{(i)})G(G^T B(\boldsymbol{x}^{(i)})G)^{-1}\right) \otimes B(\boldsymbol{x}^{(i)}).$$

The procedure is summarized in Algorithm 1. In the next section, we propose a relevant choice for the initialization G^0 of Algorithm 1. We emphasize that assembling these Km-by-Km matrices would require the storage of K^2m^2 scalars, which is obviously not affordable when K (and m) are large. In practice, we never assemble these matrices explicitly. Using the formulas

$$\widehat{H}(G)x = \left(\frac{1}{N}\sum_{i=1}^{N} A(\boldsymbol{x}^{(i)})x_{\mathrm{mat}}(G^T B(\boldsymbol{x}^{(i)})G)^{-1}\right)_{\mathrm{vec}} \tag{29}$$

$$\widehat{\Sigma}(G)x = \left(\frac{1}{N}\sum_{i=1}^{N} B(\boldsymbol{x}^{(i)})x_{\mathrm{mat}}(G^T B(\boldsymbol{x}^{(i)})G)^{-1}G^T A(\boldsymbol{x}^{(i)})G(G^T B(\boldsymbol{x}^{(i)})G)^{-1}\right)_{\mathrm{vec}}, \tag{30}$$

the matrix-vector products $x \mapsto H(G)x$ and $x \mapsto \Sigma(G)x$ are computationally tractable. In this sense, the matrices $H(G)$ and $\Sigma(G)$ are *implicit* matrices. For the calculation of $x \mapsto \widehat{\Sigma}(G)^{-1}x$, as required in (26), iterative solvers are well suited because they rely only on matrix-vector products; see [17]. Here we use a conjugate gradient solver preconditioned with the diagonal matrix containing the diagonal of $\widehat{\Sigma}(G)$.

Algorithm 1: Quasi-Newton method to maximize $G \mapsto \widehat{\mathcal{R}}(G)$.

Require: Computing the matrix-vector products $x \mapsto \widehat{H}(G)x$ and $x \mapsto \widehat{\Sigma}(G)x$ as in (29) and (30).

Data: Training sample

Input: Feature map space $\mathcal{G}_m$, initial guess $G^{(0)} \in \mathbb{R}^{K\times m}$, tolerance $\varepsilon > 0$, max iteration $K_{\max}$

Initialize $k = 0$ and stepsize $= \varepsilon + 1$

while $k < K_{\max}$ and stepsize $\geq \varepsilon$ **do**

Compute $b = \widehat{H}(G^{(k)})G^{(k)}_{\mathrm{vec}} \in \mathbb{R}^{Km}$

Solve $\widehat{\Sigma}(G^{(k)})x = b$ using preconditioned conjugate gradient

Matricize $x_{\mathrm{mat}} = (x)_{\mathrm{mat}} \in \mathbb{R}^{K\times m}$ and update $G^{(k+1/2)} = G^{(k)} - x_{\mathrm{mat}}$

Normalize $G^{(k+1)} = G^{(k+1/2)}M^{-1/2}$ with

$$M = G^{(k+1/2)T}\mathrm{Cov}(\Phi(\boldsymbol{X}))G^{(k+1/2)} \in \mathbb{R}^{m\times m}$$

Update $k \leftarrow k+1$ and stepsize $\leftarrow \|x\|$

end

Output: final iterate $G^{(k)}$

3.2 Adaptive polynomial feature map space

In the previous section we proposed an algorithm for minimizing $g \mapsto \widehat{J}(g)$ over a given feature map space $\mathcal{G}_m$, as in (15). In this section, we borrow ideas from [5, 30, 10] to construct $\mathcal{G}_m$ adaptively using multivariate polynomials.

We assume that the probability density function π of $\boldsymbol{X}$ is a product density $\pi(\boldsymbol{x}) = \pi_1(\boldsymbol{x}_1) \dots \pi_d(\boldsymbol{x}_d)$. For any $1 \leq \nu \leq d$ we denote by $\{\Phi_0^\nu, \Phi_1^\nu, \dots\}$ an orthonormal polynomial basis, with the degree of Φ_i^ν equal to i, such that

$$\int \Phi_i^\nu(x)\Phi_j^\nu(x)\pi_\nu(x)\mathrm{d}x = \delta_{ij},$$

holds for any $i, j \geq 0$. For any multi-index $\alpha = (\alpha_1, \dots, \alpha_d) \in \mathbb{N}^d$, we define the multivariate polynomial Φ_α as

$$\Phi_\alpha(\boldsymbol{x}) = \prod_{\nu=1}^{d} \Phi_{\alpha_\nu}^\nu(\boldsymbol{x}_\nu),$$

and, for a given multi-index set $\Lambda_K \subseteq \mathbb{N}^d$ of cardinality $\#\Lambda_K = K$, we introduce

$$\mathcal{G}_m^{\Lambda_K} = \left\{ \boldsymbol{x} \mapsto \begin{pmatrix} g_1(\boldsymbol{x}) \\ \vdots \\ g_m(\boldsymbol{x}) \end{pmatrix},\ g_i \in \operatorname{span}\{\Phi_\alpha; \alpha \in \Lambda_K\} \right\}. \tag{31}$$

This feature map space parametrized by Λ_K is, up to a change of notation, of the form of $\mathcal{G}_m$ in (15). The optimal multi-index set Λ_K is that which minimizes the minimum of $J(g)$ over $g \in \mathcal{G}_m^{\Lambda_K}$, meaning

$$\operatorname*{arg\,min}_{\substack{\Lambda_K \subseteq \mathbb{N}^d \\ \#\Lambda_K = K}} \min_{g \in \mathcal{G}_m^{\Lambda_K}} \widehat{J}(g). \tag{32}$$

This best K-term approximation problem is combinatorial and not tractable in practice. We propose a suboptimal solution to (32) using a greedy procedure of the form

$$\Lambda_{K+1} = \Lambda_K \cup \{\alpha_{K+1}\},$$

where $\alpha_{K+1} \in \mathbb{N}^d$ is a multi-index to determine. Suppose we are given Λ_K and that the corresponding optimal feature map

$$g_{\Lambda_K} \in \operatorname*{argmin}_{g \in \mathcal{G}_m^{\Lambda_K}} \widehat{J}(g)$$

has been computed (for instance using Algorithm 1). The optimal multi-index α_{K+1} to add would be the one which minimizes $\alpha \mapsto \widehat{J}(g_{\Lambda_K \cup \{\alpha\}})$. This would require the computation of $g_{\Lambda_K \cup \{\alpha\}}$ for many $\alpha \in \mathbb{N}^d$, which is not affordable in practice. Instead we choose the multi-index α_{K+1} as the one which yields the steepest gradient of the function $v \mapsto \widehat{J}(g_{\Lambda_K} + v\Phi_\alpha)$ around $v = 0$, meaning

$$\alpha_{K+1} \in \operatorname*{arg\,max}_{\alpha \in \mathbb{N}^d} \left\| \nabla_v\ \widehat{J}(g_{\Lambda_K} + v\Phi_\alpha)\big|_{v=0} \right\|. \tag{33}$$

The rationale behind (33) is to select the polynomial Φ_α which, once added to the feature map space $\mathcal{G}_m$, yields the best immediate improvement of $\widehat{J}(\cdot)$ when moving away from g_{Λ_K} in the direction Φ_α.

Maximization over the entire $\mathbb{N}^d$ as in (33) is not feasible in practice. A standard workaround is to search for the maximum over an arbitrary subset of $\mathbb{N}^d$ with finite cardinality. The subset



(a) Λ_K, $\mathcal{M}(\Lambda_K)$ and α_{K+1}. (b) Λ_{K+1} and $\mathcal{M}(\Lambda_{K+1})$.

Figure 2: Greedy construction of the downward closed set $\Lambda_K \subseteq \mathbb{N}^d$ with $d = 2$. Adding α_{K+1} (the cross on the left) to Λ_K (gray boxes on the left) yields Λ_{K+1} and the new reduced margin $\mathcal{M}(\Lambda_{K+1})$ (right plot).

$\{\alpha \in \mathbb{N}^d, \sum_{i=1}^d \alpha_i \leq p\}$ is commonly used, as it corresponds to the polynomials Φ_α with total degree bounded by p. However the cardinality of this subset is $\binom{d+p}{d} = \frac{(d+p)!}{p!d!}$ which can still be very large. Borrowing ideas from [30, 31], we propose an alternative strategy which relies on the notion of downward-closed sets; see [8, 10]. We assume that the set Λ_K is downward-closed, meaning that

$$\alpha \in \Lambda_K \text{ and } \alpha' \leq \alpha \;\Rightarrow\; \alpha' \in \Lambda_K, \tag{34}$$

where $\alpha' \leq \alpha$ means $\alpha'_i \leq \alpha_i$ for all $1 \leq i \leq d$. Intuitively, (34) means that Λ_K has a pyramidal shape that contains no hole. We denote by $\mathcal{M}(\Lambda_K)$ the *reduced margin* of Λ_K, defined by

$$\mathcal{M}(\Lambda_K) = \{\alpha \in \mathbb{N}^d \backslash \Lambda_K \text{ such that } \alpha - e_i \in \Lambda_K \text{ for all } 1 \leq i \leq d \text{ with } \alpha_i \neq 0\}$$

where e_i denotes the i-th canonical vector of $\mathbb{N}^d$. By construction, any set of the form $\Lambda_K \cup \{\alpha\}$ with $\alpha \in \mathcal{M}(\Lambda_K)$ remains downward closed, which is the fundamental property of the reduced margin. By searching for the new multi-index in the reduced margin of Λ_K, as in

$$\alpha_{K+1} \in \underset{\alpha \in \mathcal{M}(\Lambda_K)}{\operatorname{argmax}} \left\| \nabla_v \, \widehat{J}(g_{\Lambda_K} + v\Phi_\alpha)\big|_{v=0} \right\|,$$

we ensure that Λ_{K+1} remains downward closed. This is illustrated on Figure 2.

As pointed out in [30, 10] in the context of least-squares regression, adding multiple multi-indices at each greedy iteration could yield better performance compared to adding only one multi-index at a time. Instead of the enrichment $\Lambda_{K+1} = \Lambda_K \cup \{\alpha_{K+1}\}$, we consider the so-called *bulk chasing* procedure

$$\Lambda_{K+1} = \Lambda_K \cup \lambda_{K+1},$$

where $\lambda_{K+1} \subseteq \mathcal{M}(\Lambda_K)$ is the smallest set of multi-indices such that

$$\left(\sum_{\alpha \in \lambda_{K+1}} \left\| \nabla_v \, \widehat{J}(g_{\Lambda_K} + v\Phi_\alpha)\big|_{v=0} \right\|^2 \right) \geq \theta \left(\sum_{\alpha \in \mathcal{M}(\Lambda_K)} \left\| \nabla_v \, \widehat{J}(g_{\Lambda_K} + v\Phi_\alpha)\big|_{v=0} \right\|^2 \right), \tag{35}$$

for some parameter $0 < \theta \leq 1$. That is, λ_{K+1} contains the $\#\lambda_{K+1}$ largest values of $\|\nabla_v \, \widehat{J}(g_{\Lambda_K} + v\Phi_\alpha)\big|_{v=0}\|$ which capture a prescribed fraction θ of the norm of the gradient of J on the reduced margin. With the bulk chasing procedure we have $\#\Lambda_K \neq K$ in general.

This procedure is summarized in Algorithm 2. We choose to start the algorithm with the set $\Lambda_K = \Lambda_d = \{\alpha \in \mathbb{N}^d : \sum_{i=1}^d \alpha_i = 1\}$. This corresponds to the space of linear feature maps and,

as explained in Remark 3.3, Algorithm 1 boils down to a power iteration for which a random initialization works well. Later, we initialize Algorithm 1 by adding a row of zeros to G_{Λ_K} to account for the newly added basis terms. Notice that Algorithm 2 stops after $K_{\max}$ iterations. We will explain in Section 3.4 how to use cross validation to determine $K_{\max}$.

Algorithm 2: Construction of feature map g on a downward-closed polynomial space

Data: Training sample
Input: Intermediate dimension m, max iteration $K_{\max}$, parameter θ
Initialize $K = d$ and $\Lambda_K = \{\alpha \in \mathbb{N}^d : \sum_{i=1}^d \alpha_i = 1\}$
Compute $G_{\Lambda_K} \in \mathbb{R}^{d\times m}$ using Algorithm 1 with random initialization.
Define $g_{\Lambda_K}(\boldsymbol{x}) = G_{\Lambda_K}^T \boldsymbol{x}$
for $K = d, \dots, K_{\max} - 1$ **do**
  Compute $\|\nabla_v \widehat{J}(g_{\Lambda_K} + v\Phi_\alpha)|_{v=0}\|$ for all $\alpha \in \mathcal{M}(\Lambda_K)$
  Select λ_{K+1} as in (35)
  Update $\Lambda_{K+1} = \Lambda_K \cup \lambda_{K+1}$ and $\mathcal{G}_m^{\Lambda_{K+1}}$
  Compute $G_{\Lambda_{K+1}} \in \mathcal{G}_m^{\Lambda_{K+1}}$ using Algorithm 1 initialized with

$$G_{\Lambda_{K+1}}^{(0)} = \begin{bmatrix} G_{\Lambda_K} \\ [0, \dots, 0] \end{bmatrix} \in \mathbb{R}^{(K+1)\times m}$$

  Define $g_{\Lambda_{K+1}}(\cdot) = G_{\Lambda_{K+1}}^T \Phi(\cdot)$, where $\Phi = [\Phi_1, \dots, \Phi_{\alpha_{K+1}}] : \mathbb{R}^d \to \mathbb{R}^{K+1}$
end
Output: final iterate $g_{\Lambda_{K_{\max}}}$

Remark 3.4. The greedy procedure of Algorithm 2 can get stuck because it "doesn't see" behind the reduced margin. For instance, if a relevant index is located above $\mathcal{M}(\Lambda_K)$ and if the gradient vanishes on the reduced margin, the algorithm will never activate that index. [31] suggests a safeguard mechanism to avoid this: arbitrarily activate the most ancient index from the reduced margin every n-th iteration. In our numerical tests, however, we never needed such a safeguard mechanism.

3.3 Adaptive polynomial profile function space

In this section we assume the feature map g has been computed using Algorithm 2. We now build the profile function f in a polynomial space $\mathcal{F}_m$. As in the previous section, we propose to greedily enrich $\mathcal{F}_m$ so that the minimum of the empirical mean squared error $\widehat{\mathcal{E}}_g(f) = \frac{1}{N}\sum_{i=1}^N (u(\boldsymbol{x}^{(i)}) - f \circ g(\boldsymbol{x}^{(i)}))^2$ over $f \in \mathcal{F}_m$ is minimized. Since the gradients $u(\boldsymbol{x}^{(1)}), \dots, u(\boldsymbol{x}^{(N)})$ are available, we instead consider the gradient-enhanced empirical mean squared error,

$$\widehat{\mathcal{E}}_g^{\nabla}(f) = \frac{1}{N}\sum_{i=1}^N \Big((u(\boldsymbol{x}^{(i)}) - f \circ g(\boldsymbol{x}^{(i)}))^2 + \|\nabla u(\boldsymbol{x}^{(i)}) - \nabla f \circ g(\boldsymbol{x}^{(i)})\|^2 \Big). \tag{36}$$

Using $\widehat{\mathcal{E}}_g^{\nabla}(f)$ instead of $\widehat{\mathcal{E}}_g(f)$ is known to yield better mean squared error in the small sample regime; see [33]. This will be illustrated in the next section. Given a finite multi-index set $\Gamma_L \subseteq \mathbb{N}^m$ we

introduce

$$\mathcal{F}_m^{\Gamma_L} = \operatorname{span}\{\Psi_\alpha; \alpha \in \Gamma_L\}, \tag{37}$$

where Ψ_α denotes the α-th multivariate Hermite polynomial. These polynomials form an orthogonal basis of $L^2_{\mathcal{N}(0,I_m)}$. In the present context it would have been preferable to work with a $L^2_{g_\sharp\mu}$-orthogonal basis, but such a basis is not readily obtainable as it would require computing expensive high-dimensional integrals (e.g., for a Gram-Schmidt procedure). We justify the use of Hermite basis by the fact that, since $g(\boldsymbol{X})$ is centered and has identity covariance (recall the constraints in (14)), $\{\Psi_\alpha\}_{\alpha\in\mathbb{N}^d}$ is a relatively well conditioned basis in $L^2_{g_\sharp\mu}$. We show numerically in Section 4 that Hermite polynomials perform well.

As in the previous section, we propose to build a sub-optimal solution to the best L-term approximation problem

$$\min_{\substack{\Gamma_L\subseteq\mathbb{N}^d\\ \#\Gamma_L=L}} \min_{f\in\mathcal{F}_m^{\Gamma_L}} \widehat{\mathcal{E}}_g^{\nabla}(f)$$

by greedily constructing the multi-index set as follows: $\Gamma_{L+1} = \Gamma_L \cup \lambda_{L+1}$, where $\lambda_{L+1} \subseteq \mathcal{M}(\Gamma_L)$ is the smallest multi-index set such that

$$\left(\sum_{\alpha\in\lambda_{L+1}} \left|\frac{\mathrm{d}}{\mathrm{d}t}\widehat{\mathcal{E}}_g^{\nabla}(f_{\Gamma_L} + t\Psi_\alpha)\Big|_{t=0}\right|^2\right) \geq \theta \left(\sum_{\alpha\in\mathcal{M}(\Gamma_L)} \left|\frac{\mathrm{d}}{\mathrm{d}t}\widehat{\mathcal{E}}_g^{\nabla}(f_{\Gamma_L} + t\Psi_\alpha)\Big|_{t=0}\right|^2\right). \tag{38}$$

Here, f_{Γ_L} denotes the minimizer of $\widehat{\mathcal{E}}_g^{\nabla}(f)$ over $f \in \mathcal{F}_m^{\Gamma_L}$ and $\mathcal{M}(\Gamma_L)$ the reduced margin of Γ_L. This is summarized in Algorithm 3. Since $\widehat{\mathcal{E}}_g^{\nabla}(f)$ is quadratic in f, this algorithm corresponds to an Orthogonal Matching Pursuit (OMP) approach, as explained in the next remark.

Remark 3.5. Using the expansion $f = \sum_{l=1}^L w_l\Psi_{\alpha_l} \in \mathcal{F}_m^{\Gamma_L}$ with $\mathbf{w} = (w_1,\dots,w_L)^T \in \mathbb{R}^L$, the gradient-enhanced empirical mean squared error (36) can be written as $\widehat{\mathcal{E}}_g^{\nabla}(f) = \|y - A\mathbf{w}\|^2$, where $y \in \mathbb{R}^{N(d+1)}$ is given by

$$y = \frac{1}{\sqrt{N}}\begin{pmatrix} u(\boldsymbol{x}^{(1)}) & \dots & u(\boldsymbol{x}^{(N)}) \\ \nabla u(\boldsymbol{x}^{(1)}) & \dots & \nabla u(\boldsymbol{x}^{(N)}) \end{pmatrix}_{\mathrm{vec}}$$

and the α-th column of the matrix $A = [A_{\alpha_1}\cdots A_{\alpha_L}] \in \mathbb{R}^{N(d+1)\times L}$ is

$$A_\alpha = \frac{1}{\sqrt{N}}\begin{pmatrix} \Psi_\alpha(\boldsymbol{z}^{(1)}) & \dots & \Psi_\alpha(\boldsymbol{z}^{(N)}) \\ \nabla g(\boldsymbol{x}^{(1)})\nabla\Psi_\alpha(\boldsymbol{z}^{(1)}) & \dots & \nabla g(\boldsymbol{x}^{(N)})\nabla\Psi_\alpha(\boldsymbol{z}^{(N)}) \end{pmatrix}_{\mathrm{vec}}$$

with $\boldsymbol{z}^{(i)} = g(\boldsymbol{x}^{(i)})$. Recall that the subscript "vec" stands for the vectorization of a matrix. Thus we have $|\frac{\mathrm{d}}{\mathrm{d}t}\widehat{\mathcal{E}}_g^{\nabla}(f + t\Psi_\alpha)|_{t=0}| = |A_\alpha^T(y - Ax^L)|$, which shows that the selection procedure (38) corresponds to choosing the (nonactive) column of A_α which is most correlated with the residual $y - Ax^L$. This is similar to the OMP algorithm [40]; the difference is that, instead of seeking α in a prescribed set, Algorithm (38) seeks α in $\mathcal{M}(\Gamma_L)$, which evolves during the iteration process.

3.4 Cross-validation

Algorithms 2 and 3 need to be stopped before they begin overfitting the data. We employ the ν-fold cross-validation procedure decribed in Algorithm 4. It consists of partitioning the initial sample $\Xi = \{(\boldsymbol{x}^{(i)}, u(\boldsymbol{x}^{(i)}), \nabla u(\boldsymbol{x}^{(i)})\}_{i=1}^N$ into ν subsets Ξ_i^{train}, $i = 1,\dots,\nu$ of equal cardinality N/ν, then running the algorithms on each subset Ξ_i^{train} while monitoring the error on the corresponding test set $\Xi_i^{\mathrm{test}} = \Xi\backslash\Xi_i^{\mathrm{train}}$. The optimal number of iterations K^* (for Algorithm 2) and L^* (for Algorithm

Algorithm 3: Construction of profile function f on downward-closed polynomial space

Data: Training sample

Input: Feature map g with intermediate dimension m, max iteration $L_{\max}$, parameter θ

Initialize $\Gamma_0 = \{(0, \dots, 0)\}$

Solve the least-squares problem $f_{\Gamma_0} = \min\{\widehat{\mathcal{E}}_g^{\nabla}(f); f \in \mathcal{F}_m^{\Gamma_0}\}$

for $L = 0, \dots, L_{\max} - 1$ **do**

  Compute $|\frac{\mathrm{d}}{\mathrm{d}t}\widehat{\mathcal{E}}_g^{\nabla}(f + t\Psi_\alpha)|_{t=0}|$ for all $\alpha \in \mathcal{M}(\Gamma_L)$

  Select λ_{L+1} as in (38)

  Update $\Gamma_{L+1} = \Gamma_L \cup \lambda_{L+1}$ and $\mathcal{F}_m^{\Gamma_{L+1}}$

  Solve the least-squares problem $f_{\Gamma_{L+1}} = \min\{\widehat{\mathcal{E}}_g^{\nabla}(f); f \in \mathcal{F}_m^{\Gamma_{L+1}}\}$

end

Output: final iterate $f_{\Gamma_{L_{\max}}}$

3) are those which minimize the test error averaged over the ν folds. With these numbers in hand, we then run K^* and L^* iterations of the algorithms on the entire sample.

In Algorithm 4, we use the same sample to train both f and g. Alternatively, we can build f and g using two independent samples. We tried this alternative without obtaining significant improvement. Thus, in the context where the model u is expensive to evaluate, we recommend

training f and g on the same sample.

Algorithm 4: Learning a composed model $f \circ g \approx u$ using values and gradients of u

Data: Sample $\{(\boldsymbol{x}^{(i)}, u(\boldsymbol{x}^{(i)}), \nabla u(\boldsymbol{x}^{(i)})\}_{i=1}^{N}$
Input: Intermediate dimension m, max iteration $K_{\max}$ and $L_{\max}$, number of folds ν

Partition the data set $\Xi = \{(\boldsymbol{x}^{(i)}, u(\boldsymbol{x}^{(i)}), \nabla u(\boldsymbol{x}^{(i)})\}_{i=1}^{N}$ for cross validation
Partition Ξ into ν subsets of equal cardinality:
- **i-th test set:** Ξ_i^{test} is the i-th subset of Ξ
- **i-th training set:** $\Xi_i^{\text{train}} = \Xi \backslash \Xi_i^{\text{test}}$

Construction of the feature map
for $i = 1, \ldots, \nu$ **do**
Run $K_{\max}$ iterations of Algorithm 2 on the **i-th training set**
Store the iterates $g^{(1)}, \ldots, g^{(K_{\max})}$
Monitor the loss $\mathcal{J}_{i,j} = \widehat{J}(g^{(j)})$, $1 \leq j \leq K_{\max}$, on the **i-th test set**
end
Define K^* as the minimum of the mean $j \mapsto \frac{1}{\nu} \sum_{i=1}^{\nu} \mathcal{J}_{i,j}$
Run K^* iterations of Algorithm 2 using the **whole sample** Ξ
return feature map $g = g^{(K^*)}$

Construction of the profile
for $i = 1, \ldots, \nu$ **do**
Run $L_{\max}$ iterations of Algorithm 3 on the **i-th training set**
Store the iterates $f^{(1)}, \ldots, f^{(L_{\max})}$
Monitor the mean squared error $\mathcal{E}_{i,j} = \widehat{\mathcal{E}}_g(f^{(j)})$, $1 \leq j \leq L_{\max}$, on the **i-th test set**
end
Define L^* as the minimum of the mean $j \mapsto \frac{1}{\nu} \sum_{i=1}^{\nu} \mathcal{E}_{i,j}$
Run L^* iterations of Algorithm 3 using the **whole sample** Ξ
return profile function $f = f^{(L^*)}$

Output: Composed approximation $f \circ g$

4 Numerical examples

Source code for the algorithms above and numerical experiments below is freely available[2] so that all results presented here are entirely reproducible. Our implementation uses the toolbox *ApproximationToolbox* [3].

4.1 Isotropic function

We first consider the function $u : \mathbb{R}^d \to \mathbb{R}$ with $d = 20$ defined by

$$u(\boldsymbol{x}) = \cos(\|\boldsymbol{x}\|_2),$$

[2] `https://gitlab.inria.fr/ozahm/nonlinear-dimension-reduction-for-surrogate-modeling.git`

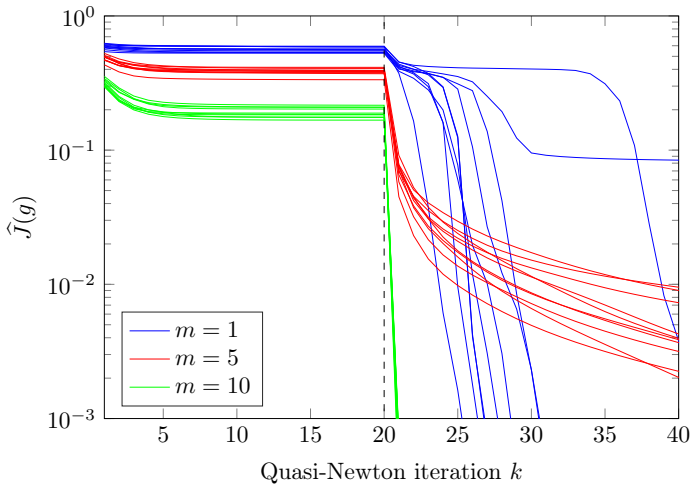


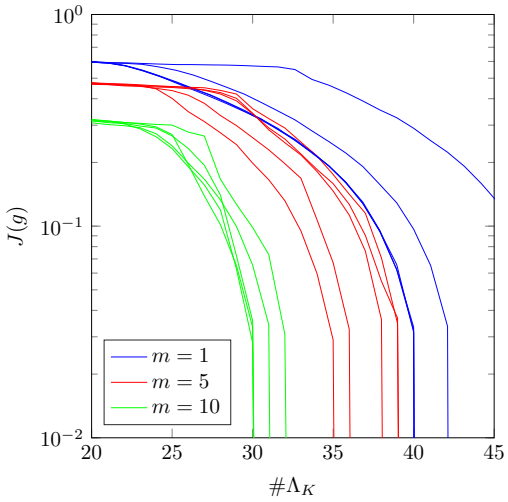
Figure 3: Isotropic function. Evolution of $\widehat{J}(g)$ during the quasi-Newton algorithm 1 using $N = 100$ gradients of u (10 different realizations). For the first 20 iterations, $\mathcal{G}_m^{\Lambda_K}$ contains linear functions only ($\#\Lambda_K = 20$). At the 21st iteration, $\mathcal{G}_m^{\Lambda_K}$ is enlarged to include all quadratic functions ($\#\Lambda_K = 20 + 210 = 230$).

and we let $\mu = \mathcal{N}(0, I_d)$ be the standard normal distribution. This function is isotropic: it cannot be well approximated by $f \circ g$ with a linear feature map g. However, if one allows g to be a quadratic polynomial, the function $g(x) = x_1^2 + \ldots + x_{20}^2 = \|\boldsymbol{x}\|_2^2$ allows one to write $u = f \circ g$ with a rather simple one-dimensional profile function, $f(z) = \cos(\sqrt{z})$.

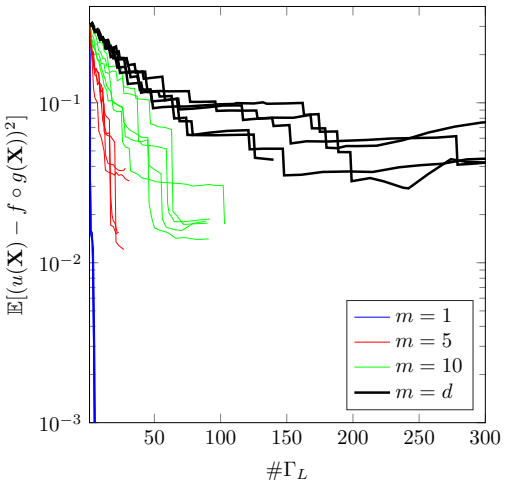
First we assess the performance of the quasi-Newton method (Algorithm 1) for the minimization of $g \mapsto \widehat{J}(g)$ over a *fixed* space of feature maps $\mathcal{G}_m$. Results are reported in Figure 3. During the first 20 iterations, $\mathcal{G}_m$ is chosen to be the space of linear feature maps; after the 21st iteration, $\mathcal{G}_m$ is enlarged to contain linear and quadratric feature maps. During the first period, we observe a rapid convergence of $J(g)$ towards a plateau which decreases with m. Once the quadratic terms are activated, $J(g)$ converges toward zero at an exponential rate. This shows the efficiency of the quasi-Newton approach in Algorithm 1 for building g on a fixed function space $\mathcal{G}_m^{\Lambda_K}$. We observe that the convergence rates are not the same for $m = 1$, $m = 5$, and $m = 10$.

Figure 4a shows the behavior of the *adaptive* Algorithm 2 for constructing a feature map g. Recall that Algorithm 2 is initialized with $\Lambda_K = \{\alpha \in \mathbb{N}^{20} : \sum_{i=1}^d \alpha_i = 1\}$, which corresponds to the space of linear feature maps. For this experiment, we enrich Λ_K with only one multi-index at a time, i.e., $\Lambda_{K+1} = \Lambda_K \cup \{\alpha_{K+1}\}$ with α_{K+1} as in (33). We observe that the algorithm is always capable of building a polynomial g such that $J(g) = 0$ with very few greedy iterations. Note that for large m, $J(g) = 0$ is attained earlier, i.e., for smaller $\#\Lambda_K$. To explain this phenomenon, Table 1 lists a few exact decompositions $u = f \circ g$, where we see that a large intermediate dimension m compensates for a small feature map space $\#\Lambda_K$.

Figure 4b shows the performance of Algorithm 3. We set the bulk chasing parameter to $\theta = 0.3$ and we run a cross-validation procedure (Algorithm 4) with $\nu = 5$ folds to determine when to stop the enrichment process. With $m = 1$, the algorithm is capable of recovering a very accurate approximation to u (error below 10^{-4}) with only $N = 100$ samples. In contrast, using the same sample, a full dimensional polynomial approximation (black curves in Figure 4b) can barely attain errors below 10^{-1}. With intermediate dimensions $m = 5$ and $m = 10$, we still outperform the full dimensional approach $d = m$, but the error does not reach 10^{-2}. This example nicely illustrates the fundamental issue of balancing the complexity between f and g:



(a) Evolution of $J(g)$ during the greedy enrichment process of Algorithm 2.



(b) Evolution of the mean squared error during the greedy Algorithm 3. The black curve $m = d$ is obtained by running Algorithm 3 with $g(x) = x$, the identity map.

Figure 4: Isotropic function. Performances of Algorithms 2 and 3 using $N = 100$ samples (5 realizations). First, we construct g using Algorithm 2 (left plot) and then, given g, we construct f using Algorithm 3 (right plot). Both $J(g)$ and $\mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2]$ are computed here on a large validation sample of size 2000.

- With $m = 1$, we obtain a complex $g \in \mathcal{G}_m^{\Lambda_K}$ with $\#\Lambda_K \geq 40$ and a simple $f \in \mathcal{F}_m^{\Gamma_L}$ with $\#\Gamma_L \leq 5$. Error is below 10^{-4}.
- With $m = 5$ or $m = 10$, we obtain a simpler $g \in \mathcal{G}_m^{\Lambda_K}$ with $30 \leq \#\Lambda_K \leq 40$ and a more complex $f \in \mathcal{F}_m^{\Gamma_L}$ with $20 \leq \#\Gamma_L \leq 100$. Error is around 2×10^{-2}.
- With $m = d$, (no dimension reduction) $g(x) = x$ is linear and $f \in \mathcal{F}_m^{\Gamma_L}$ with $\#\Gamma_L \geq 300$. Error barely falls below 10^{-1}.

Clearly, for the considered isotropic function, the optimal choice of intermediate dimension is $m = 1$. We will see in the next examples that this is not always the case.

4.2 Borehole function

Our second example is the commonly used Borehole function [39], which models water flow through a borehole. It is a function of $d = 8$ variables defined by

$$u(\boldsymbol{X}) = \frac{2\pi T_u (H_u - H_l)}{\ln(r/r_w)\left(1 + \frac{2LT_r}{\ln(r/r_w) r_w^2 K_w} + \frac{T_r}{T_l}\right)},$$

where $\boldsymbol{X}$ is a random vector in $\mathbb{R}^d$ with independent components given by

$$\begin{aligned}
X_1 = r_w &\sim \mathcal{N}(0.10, 0.0161812), & X_5 = r &\sim \log\mathcal{N}(7.71, 1.0056),\\
X_2 = T_u &\sim \mathcal{U}[63070, 115600], & X_6 = H_u &\sim \mathcal{U}[990, 1110],\\
X_3 = T_l &\sim \mathcal{U}[63.1, 116], & X_7 = H_l &\sim \mathcal{U}[700, 820],\\
X_4 = L &\sim \mathcal{U}[1120, 1680], & X_8 = K_w &\sim \mathcal{U}[9855, 12045].
\end{aligned}$$

$m = 1$	$f(z) = \cos(\sqrt{z})$	$g(\boldsymbol{x}) = (x_1^2 + \ldots + x_{20}^2)$	$\#\Lambda_K = 40$
$m = 2$	$f(z_1, z_2) = \cos(\sqrt{z_1^2 + z_2})$	$g(\boldsymbol{x}) = \begin{pmatrix} x_1 \\ x_2^2 + \ldots + x_{20}^2 \end{pmatrix}$	$\#\Lambda_K = 39$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$m = 19$	$f(z_1, \ldots, z_{19}) = \cos(\sqrt{z_1^2 + \ldots + z_{18}^2 + z_{19}})$	$g(\boldsymbol{x}) = \begin{pmatrix} x_1 \\ \vdots \\ x_{18} \\ x_{19}^2 + x_{20}^2 \end{pmatrix}$	$\#\Lambda_K = 22$
$m = 20$	$f(z_1, \ldots, z_{20}) = \cos(\sqrt{z_1^2 + \ldots + z_{20}^2})$	$g(\boldsymbol{x}) = \begin{pmatrix} x_1 \\ \vdots \\ x_{20} \end{pmatrix}$	$\#\Lambda_K = 20$

Table 1: Isotropic function. List of exact decompositions $u = f \circ g$ with polynomials $g \in \mathcal{G}_m^{\Lambda_K}$ with $\#\Lambda_K$ ranging from 40 (and $m = 1$) to 20 (and $m = 20$). This explains why, in Figure 4a, $J(g)$ drops to zero earlier in $\#\Lambda_K$ when m is large.

We first numerically illustrate Proposition 2.9. Recall that this proposition states that, given $g \in \mathcal{G}_m$, there exists a function f such that the mean squared error $\mathbb{E}[(u(\boldsymbol{X}) - f(g(\boldsymbol{X})))^2]$ is bounded by $J(g)$ multiplied by the Poincaré-type constant $\mathbb{C}(\boldsymbol{X}|\mathcal{G}_m)$. In general, $\mathbb{C}(\boldsymbol{X}|\mathcal{G}_m)$ is unknown. We build three feature maps g: a linear map, a quadratic map, and a cubic map defined as the minimizers of $\widehat{J}(g)$ over the polynomial spaces

$$\mathcal{G}_m^{\Lambda_{\text{lin}}} \quad \text{where} \quad \Lambda_{\text{lin}} = \left\{ \alpha \in \mathbb{N}^8 : 1 \leq \sum_{i=1}^{8} \alpha_i \leq 1 \right\}, \quad \#\Lambda_{\text{lin}} = 8,$$

$$\mathcal{G}_m^{\Lambda_{\text{quad}}} \quad \text{where} \quad \Lambda_{\text{quad}} = \left\{ \alpha \in \mathbb{N}^8 : 1 \leq \sum_{i=1}^{8} \alpha_i \leq 2 \right\}, \quad \#\Lambda_{\text{quad}} = 44,$$

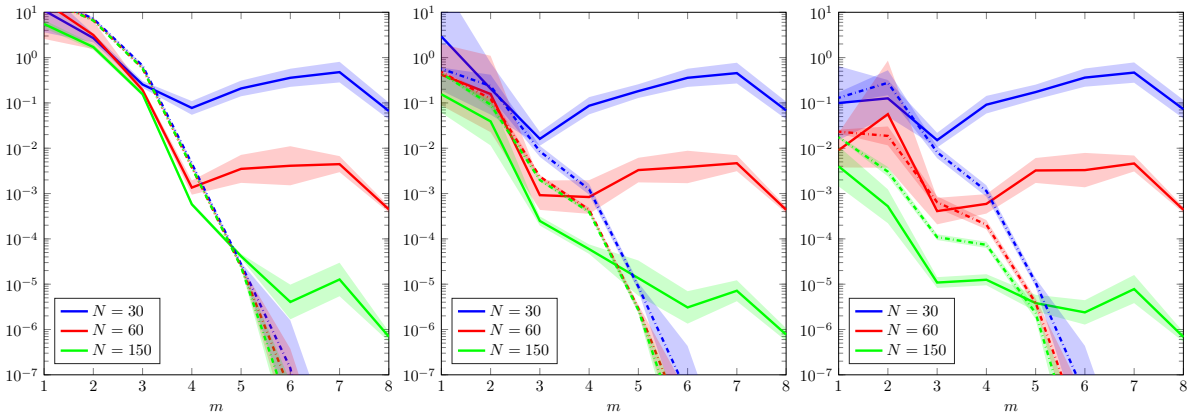
$$\mathcal{G}_m^{\Lambda_{\text{cub}}} \quad \text{where} \quad \Lambda_{\text{cub}} = \left\{ \alpha \in \mathbb{N}^8 : 1 \leq \sum_{i=1}^{8} \alpha_i \leq 3 \right\}, \quad \#\Lambda_{\text{cub}} = 164,$$

respectively. To compute these feature maps, we estimate $\widehat{J}(g)$ with $N = 30$, 60, or 150 samples. The dashed curves in Figure 5 are the resulting $J(g)$ (computed on a validation set of size $N = 2000$) as a function of m. Once g is built, we construct the profile f using Algorithm 3 on the same sample. The continuous lines in Figure 5 represent $\mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2]$ (computed on the validation set). As the sample size N increases, we obtain a better profile function f, and the mean squared error decreases until it falls below $J(g)$. We also observe that the larger m is, the higher N must be to obtain a mean squared error below $J(g)$. Domination of the mean squared error by $J(g)$ is consistent with Proposition 2.9 with a Poincaré-type constant $\mathbb{C}(\boldsymbol{X}|\mathcal{G}_m)$ that seems to be close to one for this benchmark.

In the limit $N \to \infty$, g converges towards the optimal linear/quadratic/cubic feature map while the profile function f, built adaptively in Algorithm 3, converges towards the solution of

$$\min_{f:\mathbb{R}^m \to \mathbb{R}} \mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2].$$

With a larger polynomial degree for g, the best achievable error $\min_{f:\mathbb{R}^m \to \mathbb{R}} \mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2]$ is smaller and so we obtain a better approximation $f \circ g$ to u. Notice, however, that when the mean



(a) Linear feature map (b) Quadratic feature map (c) Cubic feature map

Figure 5: Borehole. Continuous lines: mean squared error $\mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2]$, Dashed lines: cost function $J(g)$. The width of the shaded region corresponds to the standard deviation over 20 experiments. The feature map g is built by minimizing $\widehat{J}(g)$ using Algorithm 1 on samples of size $N \in \{30, 60, 150\}$. To build f, we employ Algorithm 3 on the same sample with bulk-chasing parameter $\theta = 0.3$ and a five-fold cross-validation procedure to stop the iterations.

squared error is far above $J(g)$ (typically for large m), increasing the polynomial degree of g does not significantly improve the approximation $f \circ g$. The interpretation is that if we cannot build a sufficiently accurate profile function f (either because m is too large or N is too small), there is no benefit in having a complex (i.e., high polynomial degree) feature map g.

We now build both g and f adaptively using Algorithm 4 with parameters $\theta = 0.3$ and $\nu = 5$ (from now on we use these parameters by default). Compared to the previous experiments where the polynomial degree of g was fixed, the mean squared errors shown in Figure 6a go to zero when $N \to \infty$, even for small m. Figure 6b shows the cardinalities of Λ_K and Γ_L as functions of the intermediate dimension m. We clearly see that, for small m, our adaptive algorithm builds complex feature maps and simple profile functions. For large m, it is the other way around.

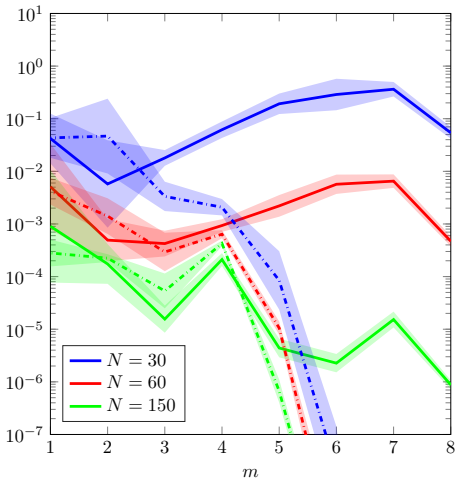
From Figure 6a, it seems that the optimal intermediate dimension m depends on N: for small sample size $N = 30$ or $N = 60$, the best intermediate dimension is $m = 2$ or $m = 3$. For $N = 150$, however, one clearly obtains better results with $m = d$, meaning without dimension reduction, i.e., $u(x) \approx f(x)$ with $g(x) = x$.

4.3 Composed function

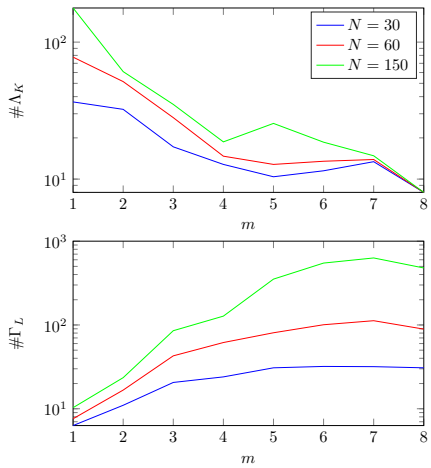
We consider now the benchmark introduced in [18] defined as a deep composition of functions. We consider the function u of $d = 16$ variables defined by

$$u(x) = h\Big(h\big(h(h(x_1, x_2), h(x_3, x_4)), h(h(x_5, x_6), h(x_7, x_8))\big),$$
$$h\big(h(h(x_9, x_{10}), h(x_{11}, x_{12})), h(h(x_{13}, x_{14}), h(x_{15}, x_{16}))\big)\Big),$$

where $h(s, t) = 9^{-1}(1 + st)^2$ and we let $\boldsymbol{X}$ be the random vector with uniform measure on $[-1, 1]^{16}$. This function u is a polynomial (as a composition of polynomials) and can readily be written as $u = f \circ g$ for $m = 2, 4, 8$ with polynomials f and g.



(a) Gradient-enhanced construction of f



(b) Mean cardinality of Λ_K (top) and of Γ_L (bottom)

Figure 6: Borehole. Same settings as for Figure 5 but with a feature map g built using the adaptive Algorithm 2. The plots on the right show the complexity of g and f through the cardinalities of Λ_K and Γ_L, respectively (mean over 20 experiments).

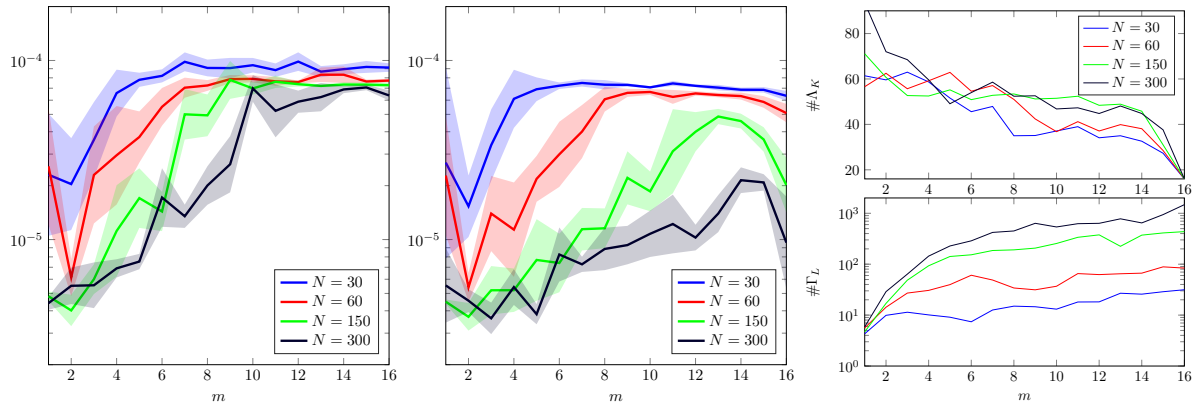
Numerical results are reported in Figure 7. For each choice of N and m, after constructing the feature map g via Algorithm 2 and the cross-validation procedure in the first half of Algorithm 4, we illustrate the benefits of the gradient-enhanced construction of the profile function f by building it either with gradient-free least squares (i.e., by minimizing $\widehat{\mathcal{E}}_g(f) = \frac{1}{N}\sum_{i=1}^N (u(\boldsymbol{x}^{(i)}) - f \circ g(\boldsymbol{x}^{(i)}))^2$) or with gradient-enhanced least squares (i.e., by minimizing $\widehat{\mathcal{E}}_g^{\nabla}(f)$ in (36)). For large m, the gradient-enhanced approach clearly outperforms the gradient-free approach, but for small m, both approaches perform equally. It seems that, for small m, the profile can be estimated accurately using evaluations of $u(\boldsymbol{x}^{(i)})$ only. Since gradients are needed to construct g regardless, our recommendation is always to use the gradient-enhanced approach to construct f, as it makes better use of the available information.

For this benchmark, it seems that $m = 2$ is the best intermediate dimension for the considered range of sample sizes N. With this choice, the mean squared error can be reduced by around a factor of 10 over a full-dimensional function approximation scheme that simply uses $g = \mathrm{Id}$ with the same sample.

4.4 Resonance frequency of a bridge

Our last numerical experiment is a PDE-based model where the quantity of interest $u(\boldsymbol{x})$ is the smallest resonance frequency of a 2D structure which has the shape of a bridge, as shown in Figure 8. Here, $\boldsymbol{x}$ parameterizes the Young modulus field of the structure. An important feature of this problem is that, while it relies on a complex numerical model, one can evaluate the gradient $\nabla u(\boldsymbol{x})$ with the same computational cost as that of an evaluation of $u(\boldsymbol{x})$, as we shall explain below.

To model the structure, we consider a linear elasticity problem in two spatial dimensions under plane stress assumption. After finite element discretization, the smallest resonance frequency $u(\boldsymbol{x})$



(a) Gradient-free construction of f

(b) Gradient-enhanced construction of f

(c) Mean cardinality of Λ_K (top) and of Γ_L (bottom).

Figure 7: Composed function. Mean squared error $\mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2]$ (computed on a validation set of size 1000) where g and f obtained by Algorithm 4 ($\theta = 0.3$ and $\nu = 5$). The line (resp. the width of the shades) corresponds to the mean (resp. the variance) over 20 experiments. Figure 7a: f is built by minimizing the gradient-free mean square $\widehat{\mathcal{E}}_g(f) = \frac{1}{N}\sum_{i=1}^N (u(\boldsymbol{x}^{(i)}) - f \circ g(\boldsymbol{x}^{(i)}))^2$. Figure 7b: f is built by minimizing by minimizing $\widehat{\mathcal{E}}_g^{\nabla}(f)$, see (36). Figure 7c: cardinalities of Λ_K and of Γ_L (with the gradient-enhanced construction of f).

is defined as the minimum of a Rayleigh quotient

$$u(\boldsymbol{x}) = \min_{v \in \mathbb{R}^n} \frac{v^T K(\boldsymbol{x}) v}{v^T M v},$$

where $K(\boldsymbol{x}) \in \mathbb{R}^{n \times n}$ and $M \in \mathbb{R}^{n \times n}$ are the stiffness and the mass matrices given by

$$K_{ij}(\boldsymbol{x}) = \int_\Omega \left\langle \frac{E(\boldsymbol{x})}{1+\nu}\varepsilon(\phi_i) + \frac{\nu E(\boldsymbol{x})}{1-\nu^2}\,\mathrm{trace}(\varepsilon(\phi_i))I_2, \varepsilon(\phi_j)\right\rangle_{\mathrm{F}} \mathrm{d}\Omega,$$
$$M_{ij} = \int_\Omega \langle \phi_i, \phi_j \rangle \,\mathrm{d}\Omega.$$

Here, $n = 960$ is the number of nodes in the finite element mesh, $\phi_i : \Omega \to \mathbb{R}^2$ is the i-th finite element function, $\varepsilon(v) = \frac{1}{2}(\nabla v + \nabla v^T) \in \mathbb{R}^{2\times 2}$ is the strain tensor, $\langle \cdot, \cdot \rangle_F$ is the Frobenius scalar product in $\mathbb{R}^{2\times 2}$, and $\langle \cdot, \cdot \rangle$ the canonical scalar product in $\mathbb{R}^2$. The Poisson coefficient is set to $\nu = 0.3$ and the Young modulus field $E(\boldsymbol{x}) : \Omega \to \mathbb{R}$ is parameterized by a $d = 32$-dimensional parameter $\boldsymbol{x} \in \mathbb{R}^d$ as follows,

$$E(\boldsymbol{x}) = \exp\left(\sum_{i=1}^{32} x_i \sqrt{\sigma_i}\psi_i\right),$$

where $\psi_i : \Omega \to \mathbb{R}$ and σ_i are the i-th leading eigenfunctions and eigenvalues of the Gaussian kernel $c(s,t) = \sqrt{5}\exp(-\|s-t\|_2^2/20)$. We endow the parameter $\boldsymbol{X}$ with the standard normal distribution on $\mathbb{R}^{32}$.

We denote by

$$v(\boldsymbol{x}) = \operatorname*{argmin}_{v \in \mathbb{R}^n} \frac{v^T K(\boldsymbol{x}) v}{v^T M v},$$

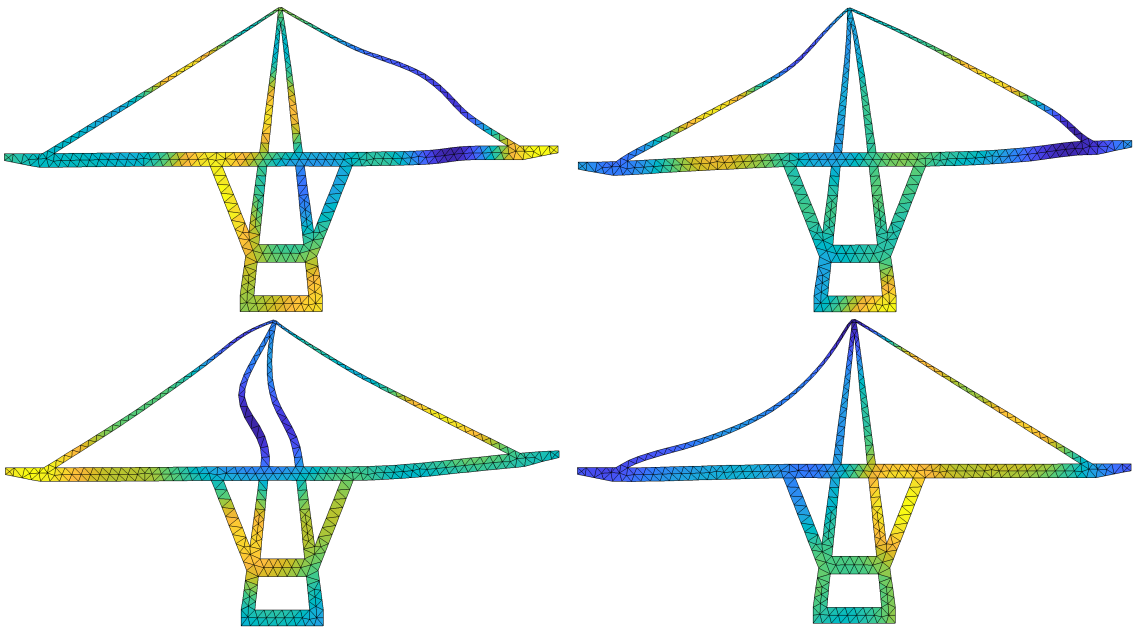


Figure 8: Resonance frequency of a bridge. Four realizations of the Young modulus field $E(\boldsymbol{X})$ (color of the elements) and the associated resonance mode $v(\boldsymbol{X})$ (displacement of the mesh).

the minimizer of the Rayleigh quotient (i.e., the eigenvector associated to the eigenvalue/frequency $u(\boldsymbol{x})$). The i-th component of $\nabla u(\boldsymbol{x}) = (\partial_{x_1} u(\boldsymbol{x}), \cdots, \partial_{x_d} u(\boldsymbol{x}))$ can be written as

$$\partial_{x_i} u(\boldsymbol{x}) = \frac{v(\boldsymbol{x})^T \big(\partial_{x_i} K(\boldsymbol{x})\big) v(\boldsymbol{x})}{v(\boldsymbol{x})^T M v(\boldsymbol{x})}. \tag{39}$$

To show this, let us write $u(\boldsymbol{x}) = R(v(\boldsymbol{x}), \boldsymbol{x})$ where $R(v, x) = \frac{v^T K(\boldsymbol{x}) v}{v^T M v}$ is the Rayleigh quotient. By definition of $v(\boldsymbol{x})$ we have $\nabla_v R(v(\boldsymbol{x}), \boldsymbol{x}) = 0$ so that a chain rule derivative yields $\partial_{x_i} u(\boldsymbol{x}) = \nabla_v R(v(\boldsymbol{x}), \boldsymbol{x})^T \partial_{x_i} v(\boldsymbol{x}) + \partial_{x_i} R(v(\boldsymbol{x}), \boldsymbol{x}) = \partial_{x_i} R(v(\boldsymbol{x}), \boldsymbol{x})$, which is (39). By definition of $E(\boldsymbol{x})$ and $K(\boldsymbol{x})$, the matrix $\partial_{x_i} K(\boldsymbol{x})$ is given by

$$\partial_{x_i} K_{kl}(\boldsymbol{x}) = \int_\Omega \sqrt{\sigma_i} \psi_i \left\langle \frac{E(\boldsymbol{x})}{1+\nu} \varepsilon(\phi_k) + \frac{\nu E(\boldsymbol{x})}{1-\nu^2} \operatorname{trace}(\varepsilon(\phi_k)) I_2, \varepsilon(\phi_l) \right\rangle_{\mathrm{F}} \mathrm{d}\Omega.$$

The cost of assembling $\partial_{x_i} K$ for $1 \leq i \leq d$ is negligible compared to the cost of computing the eigenmode $v(\boldsymbol{x})$, which requires an expensive inverse power iteration method. In other words, once $v(\boldsymbol{x})$ is computed, one can evaluate both $u(\boldsymbol{x})$ and $\nabla u(\boldsymbol{x})$ almost for free.

In Table 2 we report the performance of Algorithm 4 on this benchmark, for a sample size $N = 100$ and a range of values of m. The best performance is obtained with an intermediate dimension of $m = 3$. For $m = 8$ or $m = 16$, the mean squared error is slightly higher than for $m = d$, meaning when we don't reduce the dimension. As before, we observe that a small intermediate dimension m yields complex feature maps g (i.e., large $\#\Lambda_K$) and simple profiles f (i.e., small $\#\Gamma_L$).

	$m=1$	$m=2$	$m=3$	$m=4$	$m=6$	$m=8$	$m=16$	$m=d=32$
Mean$\times 10^{12}$	1.6	1.5	**1.1**	1.2	1.3	1.5	1.6	1.4
Std$\times 10^{12}$	0.80	0.69	**0.22**	0.24	0.28	0.83	0.39	0.43
$\#\Lambda_K$	148 ($\pm$64)	129 ($\pm$45)	91 ($\pm$21)	80 ($\pm$23)	64 ($\pm$16)	57 ($\pm$9)	51 ($\pm$1)	32 ($\pm$0)
$\#\Gamma_L$	5 ($\pm$1)	8 ($\pm$1)	11 ($\pm$1)	15 ($\pm$3)	24 ($\pm$7)	44 ($\pm$24)	133 ($\pm$102)	102 ($\pm$70)

Table 2: Bridge. Mean and standard deviation (std) of the mean squared error $\mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2]$ over 20 experiments, where g and f are constructed using Algorithm 4 with $N = 100$ samples. The error $\mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2]$ is computed on a (fixed) validation set of size 1000. The last two lines of the table give the mean($\pm$ std) of the cardinalities $\#\Lambda_K$ and $\#\Gamma_L$, which represent the complexity of g and f, respectively.

5 Conclusion

We have proposed and analyzed a novel framework for the dimension reduction of multivariate functions. Our approach relies on gradient evaluations of the model $u : \mathbb{R}^d \to \mathbb{R}$ and is a two-step procedure. First, we build a feature map $g : \mathbb{R}^d \to \mathbb{R}^m$ in a function space $\mathcal{G}_m$ by aligning the Jacobian of g with the gradients of u. Second, we build a profile function $f : \mathbb{R}^m \to \mathbb{R}$ by minimizing the mean squared error between u and $f \circ g$. We prove that having a finite Poincaré constant $\mathbb{C}(\boldsymbol{X}|\mathcal{G}_m)$ ensures good theoretical properties of the feature map—namely that the objective used to identify g bounds the L^2 error between u and its approximation. The Poincaré constant depends both on the probability measure of the inputs $\boldsymbol{X}$ and on the feature space $\mathcal{G}_m$. In practice we observe good approximation performance using polynomial spaces $\mathcal{G}_m$, constructed via a greedy adaptive procedure, but we cannot easily check that $\mathbb{C}(\boldsymbol{X}|\mathcal{G}_m) < \infty$ for this case. Indeed, theoretically guaranteeing that $\mathbb{C}(\boldsymbol{X}|\mathcal{G}_m) < \infty$ for a computationally feasible space of nonlinear feature maps $\mathcal{G}_m$ remains a challenge.

Our numerical experiments also illustrate the role of the intermediate dimension m in this setting. It is natural to ask what is the *intrinsic* intermediate dimension m of a model u? From a theoretical perspective, we argue that this question is void without specifying a function class $\mathcal{G}_m$ for g. For instance, we can talk about the *linear* or *quadratic* intrinsic intermediate dimension of u as the smallest m such that there exists a linear or a quadratic g so that the error $\mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2]$ is less than a prescribed tolerance for some $f : \mathbb{R}^m \to \mathbb{R}$. The OMP-type algorithm we propose, which adapts the complexity of $\mathcal{G}_m$ to the sample size, then makes the interpretation of m more complicated.

A useful alternative question is how to optimally select the intermediate dimension m in practice? For now, we have no way to select it *a priori*. In our numerical tests, we run the algorithm for all possible values of $m = 1, \ldots, d$ and select the intermediate dimension which yields the lowest cross-validation error. We have observed that the intermediate dimension which yields the smallest reconstruction error depends on the sample size N: for instance, in the small sample size regime, an intermediate dimension of $m = 2$ or 3 might yield better approximation while, in the large sample size regime, no dimension reduction, i.e., $m = d$, could be a better choice. This trend depends very much on the target function u, and we show examples where an intermediate value of m is best over a range of sample sizes.

The minimization of the function $J(g)$ turns out to be quite a challenging task. While the quasi-Newton method proposed here is generally effective, recent work [24] may offer a novel optimization perspective to address the essential problem of minimizing sums of generalized Rayleigh quotients.

Another interesting direction motivated by the present work is the recursive construction of approximations of the form $f_k \circ f_{k-1} \circ \ldots \circ f_1$, where each f_i is built using gradients of u. This

composition is related to deep neural network architectures for function approximation, and may offer a perspective on the choice of latent space and internal dimension in such methods.

Acknowledgment

The authors gratefully acknowledge support from the Inria associate team UNQUESTIONABLE. CP and OZ also acknowledge support from CIROQUO consortium. DB and YMM also acknowledge support from the US Department of Energy, Office of Advanced Scientific Computing Research, AEOLUS project.

References

[1] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*, Princeton University Press, 2009.

[2] K. P. Adragni and R. D. Cook, *Sufficient dimension reduction and prediction in regression*, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 367 (2009), pp. 4385–4405.

[3] N. Anthony, G. Erwan, and G. Loic, *Approximationtoolbox*, Feb. 2020.

[4] D. Bakry, F. Barthe, P. Cattiaux, A. Guillin, et al., *A simple proof of the poincaré inequality for a large class of probability measures*, Electronic Communications in Probability, 13 (2008), pp. 60–66.

[5] A. Beck and Y. C. Eldar, *Sparsity constrained nonlinear optimization: Optimality conditions and algorithms*, SIAM Journal on Optimization, 23 (2013), pp. 1480–1509.

[6] S. Boucheron, G. Lugosi, and P. Massart, *Concentration inequalities: A nonasymptotic theory of independence*, Oxford university press, 2013.

[7] M. C. Brennan, D. Bigoni, O. Zahm, A. Spantini, and Y. Marzouk, *Greedy inference with structure-exploiting lazy maps*, arXiv preprint arXiv:1906.00031, (2020).

[8] A. Chkifa, A. Cohen, and C. Schwab, *Breaking the curse of dimensionality in sparse polynomial approximation of parametric pdes*, Journal de Mathématiques Pures et Appliquées, 103 (2015), pp. 400–428.

[9] A. Cohen, I. Daubechies, R. DeVore, G. Kerkyacharian, and D. Picard, *Capturing ridge functions in high dimensions from point queries*, Constructive Approximation, 35 (2012), pp. 225–243.

[10] A. Cohen and G. Migliorati, *Multivariate approximation in downward closed polynomial spaces*, in Contemporary Computational Mathematics-A celebration of the 80th birthday of Ian Sloan, Springer, 2018, pp. 233–282.

[11] P. G. Constantine, *Active subspaces: Emerging ideas for dimension reduction in parameter studies*, SIAM, 2015.

[12] P. G. Constantine, E. Dow, and Q. Wang, *Active subspace methods in theory and practice: applications to kriging surfaces*, SIAM Journal on Scientific Computing, 36 (2014), pp. A1500–A1524.

[13] R. D. Cook and S. Weisberg, *Discussion of sliced inverse regression for dimension reduction*, Journal of the American Statistical Association, 86 (1991), pp. 328–332.

[14] T. Cui and O. Zahm, *Data-free likelihood-informed dimension reduction of bayesian inverse problems*, (2020).

[15] J. E. Dennis, Jr and J. J. Moré, *Quasi-newton methods, motivation and theory*, SIAM review, 19 (1977), pp. 46–89.

[16] M. Fornasier, K. Schnass, and J. Vybiral, *Learning functions of few arbitrary linear parameters in high dimensions*, Foundations of Computational Mathematics, 12 (2012), pp. 229–262.

[17] G. H. Golub and C. F. Van Loan, *Matrix computations*, vol. 3, JHU press, 2013.

[18] E. Grelier, A. Nouy, and M. Chevreuil, *Learning with tree-based tensor formats*, arXiv preprint arXiv:1811.04455, (2018).

[19] A. Griewank et al., *On automatic differentiation*, Mathematical Programming: recent developments and applications, 6 (1989), pp. 83–107.

[20] J. M. Hokanson and P. G. Constantine, *Data-driven polynomial ridge approximation using variable projection*, SIAM Journal on Scientific Computing, 40 (2018), pp. A1566–A1589.

[21] E. Kokiopoulou, J. Chen, and Y. Saad, *Trace optimization and eigenproblems in dimension reduction methods*, Numerical Linear Algebra with Applications, 18 (2011), pp. 565–602.

[22] S. G. Krantz and H. R. Parks, *The implicit function theorem: history, theory, and applications*, Springer Science & Business Media, 2012.

[23] R. R. Lam, O. Zahm, Y. M. Marzouk, and K. E. Willcox, *Multifidelity dimension reduction via active subspaces*, SIAM Journal on Scientific Computing, 42 (2020), pp. A929–A956.

[24] J. B. Lasserre, V. Magron, S. Marx, and O. Zahm, *Minimizing rational functions: a hierarchy of approximations via pushforward measures*, arXiv preprint arXiv:2012.05793, (2020).

[25] C. Lataniotis, S. Marelli, and B. Sudret, *Extending classical surrogate modeling to high dimensions through supervised dimensionality reduction: a data-driven approach*, International Journal for Uncertainty Quantification, 10 (2020).

[26] L. Laurent, R. Le Riche, B. Soulier, and P.-A. Boucard, *An overview of gradient-enhanced metamodels with applications*, Archives of Computational Methods in Engineering, 26 (2019), pp. 61–106.

[27] K.-Y. Lee, B. Li, F. Chiaromonte, et al., *A general theory for nonlinear sufficient dimension reduction: Formulation and estimation*, Annals of Statistics, 41 (2013), pp. 221–249.

[28] B. Li, *Sufficient dimension reduction: Methods and applications with R*, CRC Press, 2018.

[29] K.-C. Li, *Sliced inverse regression for dimension reduction*, Journal of the American Statistical Association, 86 (1991), pp. 316–327.

[30] G. Migliorati, *Adaptive polynomial approximation by means of random discrete least squares*, in Numerical Mathematics and Advanced Applications-ENUMATH 2013, Springer, 2015, pp. 547–554.

[31] ——, *Adaptive approximation by optimal weighted least-squares methods*, SIAM Journal on Numerical Analysis, 57 (2019), pp. 2217–2245.

[32] M. T. Parente, J. Wallin, B. Wohlmuth, et al., *Generalized bounds for active subspaces*, Electronic Journal of Statistics, 14 (2020), pp. 917–943.

[33] J. Peng, J. Hampton, and A. Doostan, *On polynomial chaos expansion via gradient-enhanced 1-minimization*, Journal of Computational Physics, 310 (2016), pp. 440–458.

[34] A. Pinkus, *Ridge functions*, vol. 205, Cambridge University Press, 2015.

[35] R.-E. Plessix, *A review of the adjoint-state method for computing the gradient of a functional with geophysical applications*, Geophysical Journal International, 167 (2006), pp. 495–503.

[36] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Global sensitivity analysis: the primer*, John Wiley & Sons, 2008.

[37] P. Scheiblechner, *On the complexity of deciding connectedness and computing betti numbers of a complex algebraic variety*, Journal of Complexity, 23 (2007), pp. 359–379.

[38] G. W. Stewart, *Matrix perturbation theory*, (1990).

[39] S. Surjanovic and D. Bingham, *Virtual library of simulation experiments*, 2013.

[40] J. A. Tropp and A. C. Gilbert, *Signal recovery from random measurements via orthogonal matching pursuit*, IEEE Transactions on information theory, 53 (2007), pp. 4655–4666.

[41] C. Villani, *Optimal transport: old and new*, vol. 338, Springer Science & Business Media, 2008.

[42] X. Wang, L. Wang, and Y. Xia, *An efficient global optimization algorithm for maximizing the sum of two generalized rayleigh quotients*, Computational and Applied Mathematics, 37 (2018), pp. 4412–4422.

[43] H.-M. Wu, *Kernel sliced inverse regression with applications to classification*, Journal of Computational and Graphical Statistics, 17 (2008), pp. 590–610.

[44] Y.-R. Yeh, S.-Y. Huang, and Y.-J. Lee, *Nonlinear dimension reduction with kernel sliced inverse regression*, IEEE transactions on Knowledge and Data Engineering, 21 (2008), pp. 1590–1603.

[45] O. Zahm, P. G. Constantine, C. Prieur, and Y. M. Marzouk, *Gradient-based dimension reduction of multivariate vector-valued functions*, SIAM Journal on Scientific Computing, 42 (2020), pp. A534–A558.

[46] O. Zahm, T. Cui, K. Law, A. Spantini, and Y. Marzouk, *Certified dimension reduction in nonlinear bayesian inverse problems*, arXiv preprint arXiv:1807.03712, (2018).

[47] G. Zhang, J. Zhang, and J. Hinkle, *Learning nonlinear level sets for dimensionality reduction in function approximation*, in Advances in Neural Information Processing Systems, 2019, pp. 13199–13208.

[48] L.-H. Zhang, *On optimizing the sum of the rayleigh quotient and the generalized rayleigh quotient on the unit sphere*, Computational Optimization and Applications, 54 (2013), pp. 111–139.

[49] ——, *On a self-consistent-field-like iteration for maximizing the sum of the rayleigh quotients*, Journal of Computational and Applied Mathematics, 257 (2014), pp. 14–28.

A Link with the loss function introduced in [47]

As in Example 2.5, let $\phi : \mathcal{X} \to \mathcal{X}$ be a C^1-diffeomorphism and let $g : \mathcal{X} \to \mathbb{R}^m$ be a feature map defined by $g(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \dots, \phi_m(\boldsymbol{x}))$. In [47], the diffeomorphism ϕ is built by minimizing the loss function

$$\mathcal{L}_\omega(\phi) \coloneqq \mathbb{E}\left[\sum_{i=1}^{d} \omega_i \left\langle \frac{\nabla\phi_i(\boldsymbol{X})}{\|\nabla\phi_i(\boldsymbol{X})\|}, \nabla u(\boldsymbol{X}) \right\rangle^2\right],$$

where $\omega = (\omega_1, \dots, \omega_d) \in \mathbb{R}^d_{\geq 0}$ are non-negative weights which are arbitrarily chosen. To link this loss function with the proposed cost function $J(g)$, let us assume that the orthogonality condition

$$\nabla\phi_i(\boldsymbol{x})^T \nabla\phi_j(\boldsymbol{x}) = 0, \tag{40}$$

holds for any $i \neq j$ and for any $\boldsymbol{x} \in \mathcal{X}$. Under this assumption, the cost function $J(g)$ can be written as

$$\begin{aligned} J(g) &= \mathbb{E}\left[\left\|(I_d - \Pi_{\mathrm{range}(\nabla g(\boldsymbol{X})^T)})\nabla u(\boldsymbol{X})\right\|_2^2\right] \\ &\overset{(40)}{=} \mathbb{E}\left[\sum_{i=m+1}^{d} \left\langle \frac{\nabla\phi_i(\boldsymbol{X})}{\|\nabla\phi_i(\boldsymbol{X})\|}, \nabla u(\boldsymbol{X}) \right\rangle^2\right] \\ &= \mathcal{L}_\omega(\phi), \end{aligned}$$

where the last equality is obtained by letting

$$\omega = (\underbrace{0, \dots, 0}_{m \text{ times}}, \underbrace{1, \dots, 1}_{d-m \text{ times}}).$$

In [47], the loss function $\mathcal{L}_\omega(\phi)$ is used without ensuring the orthogonality condition (40) and no theoretical justification is provided. For instance, without condition (40), it is unclear whether $\mathcal{L}_\omega(\phi) = 0$ implies $u(\boldsymbol{x}) = f \circ g(\boldsymbol{x})$ or, more critically, if $u(\boldsymbol{x}) = f \circ g(\boldsymbol{x})$ implies $\mathcal{L}_\omega(\phi) = 0$.

B Proof of Proposition 3.2

We use the notation $M_{\mathrm{sym}} = (M + M^T)/2$ for the symmetric part of a square matrix M. For any $\|\delta G\| \leq \varepsilon$ we can write

$$(G + \delta G)^T A(\boldsymbol{X})(G + \delta G) = G^T A(\boldsymbol{X}) G + 2(\delta G^T A(\boldsymbol{X}) G)_{\mathrm{sym}} + \mathcal{O}(\|\delta G\|^2),$$

and

$$\begin{aligned}
&\left((G+\delta G)^T B(\boldsymbol{X})(G+\delta G)\right)^{-1} \\
&\quad = \left(G^T B(\boldsymbol{X})G + 2(\delta G^T B(\boldsymbol{X})G)_{\mathrm{sym}} + \mathcal{O}(\|\delta G\|^2)\right)^{-1} \\
&\quad = (G^T B(\boldsymbol{X})G)^{-1} - 2(G^T B(\boldsymbol{X})G)^{-1}(\delta G^T B(\boldsymbol{X})G)_{\mathrm{sym}}(G^T B(\boldsymbol{X})G)^{-1} + \mathcal{O}(\|\delta G\|^2).
\end{aligned}$$

Multiplying the two above quantities yields

$$\begin{aligned}
&\Big((G+\delta G)^T A(\boldsymbol{X})(G+\delta G)\Big)\Big((G+\delta G)^T B(\boldsymbol{X})(G+\delta G)\Big)^{-1} \\
&\quad = (G^T A(\boldsymbol{X})G)(G^T B(\boldsymbol{X})G)^{-1} + 2(\delta G^T A(\boldsymbol{X})G)_{\mathrm{sym}}(G^T B(\boldsymbol{X})G)^{-1} \\
&\quad - 2(G^T A(\boldsymbol{X})G)(G^T B(\boldsymbol{X})G)^{-1}(\delta G^T B(\boldsymbol{X})G)_{\mathrm{sym}}(G^T B(\boldsymbol{X})G)^{-1} + \mathcal{O}(\|\delta G\|^2).
\end{aligned}$$

Taking the expectation of the trace yields

$$\begin{aligned}
&\mathcal{R}(G+\delta G) = \mathcal{R}(G) + \mathbb{E}\left[\mathrm{trace}\left(2\delta G^T A(\boldsymbol{X})G(G^T B(\boldsymbol{X})G)^{-1}\right)\right] \\
&- \mathbb{E}\left[\mathrm{trace}\left(2(G^T A(\boldsymbol{X})G)(G^T B(\boldsymbol{X})G)^{-1}(\delta G^T B(\boldsymbol{X})G)(G^T B(\boldsymbol{X})G)^{-1}\right)\right] + \mathcal{O}(\|\delta G\|^2).
\end{aligned}$$

Here we used the fact that $\mathrm{trace}(M_{\mathrm{sym}}S) = \mathrm{trace}(MS)$ holds for any square matrix M and any symmetric matrix S. Using the notation $\langle M, N\rangle = \mathrm{trace}(MN^T)$, we can write $\mathcal{R}(G+\delta G) = \mathcal{R}(G) + \langle \nabla\mathcal{R}(G), \delta G\rangle + \mathcal{O}(\|\delta G\|^2)$ where

$$\begin{aligned}
\nabla\mathcal{R}(G) = &\ 2\mathbb{E}\left[A(\boldsymbol{X})G(G^T B(\boldsymbol{X})G)^{-1}\right] \\
&- 2\mathbb{E}\left[B(\boldsymbol{X})G(G^T B(\boldsymbol{X})G)^{-1}G^T A(\boldsymbol{X})G(G^T B(\boldsymbol{X})G)^{-1}\right].
\end{aligned}$$

This shows that $\mathcal{R}(\cdot)$ is differentiable at G. Finally, the expression (22) of $\nabla\mathcal{R}(G)$ is obtained by using the definitions of $H(G)$ and $\Sigma(G)$ (see (23) and (24)) and by using the fact that $(S_1 G S_2)_{\mathrm{vec}} = (S_2 \otimes S_1)G_{\mathrm{vec}}$ for any symmetric matrices S_1, S_2. Both $H(G)$ and $\Sigma(G)$ are symmetric positive semidefinite, as the expectations of the Kronecker products of symmetric positive semidefinite matrices.