LINEAR/RIDGE EXPANSIONS: ENHANCING LINEAR APPROXIMATIONS BY RIDGE FUNCTIONS

CONSTANTIN GREIF, PHILIPP JUNK, AND KARSTEN URBAN

Abstract. We consider approximations formed by the sum of a linear combination of given functions enhanced by ridge functions – a Linear/Ridge expansion. For an explicitly or implicitly given function, we reformulate finding a best Linear/Ridge expansion in terms of an optimization problem. We introduce a particle grid algorithm for its solution. Several numerical results underline the flexibility, robustness and efficiency of the algorithm.

One particular source of motivation is model reduction of parameterized transport or wave equations. We show that the particle grid algorithm is able to produce a Linear/Ridge expansion as an efficient nonlinear model reduction.

1. Introduction

Many (numerical) approximations rely on linear approximation schemes. For some $f \in H$, H a normed space, one seeks for a possibly good finite-dimensional subspace $H_N \subset H$ and an approximation $f_N \in H_N$ of f. Examples include finite element, finite volume, spectral or discontinuous Galerkin methods. In many cases, such linear schemes work very well, in particular if the error of an approximation scheme can be shown to be bounded in terms of the error of the best approximation in H_N (e.g. by the famous Ceá lemma, [7, 28]). If then the error of the best approximation decays fast (e.g. by using a Clément-type operator, [8]), one obtains an efficient (numerical) approximation.

Of course, one would want to determine a "best-possible" approximation. For linear approximation schemes, the worst best possible error is known as the *Kolmogov N-width* $d_N(\mathcal{F})$ which is defined for a class $\mathcal{F} \subset H$ of elements as ([18, 24])

$$d_N(\mathcal{F}) := \inf_{H_N \subset H; \dim(H_N)=N} \sup_{f \in \mathcal{F}} \inf_{f_N \in H_N} \|f - f_N\|_H.$$

The class $\mathcal{F}$ could e.g. be a smoothness class (a Sobolev or Besov space) or a set of solutions for certain problems with different data (e.g. a parametric partial differential equation (PPDE) for different parameter values, see below).

The question if $d_N(\mathcal{F})$ decays "fast" as $N \to \infty$ (e.g. $d_N(\mathcal{F}) = \mathcal{O}(e^{-N})$) or "slow" (e.g. $d_N(\mathcal{F}) \simeq N^{-s}$, $0 < s < 1$) typically depends on some measure of smoothness of the elements of $\mathcal{F}$, for example the Besov regularity. If $\mathcal{F}$ is given as a set of solutions of a problem (such as a PPDE), then the decay of $d_N(\mathcal{F})$ is a property of the *problem* at hand (not its discretization). This means that linear approximation schemes are not appropriate for problems with a poor decay of the N-width. Hence, other schemes are needed, which are necessarily nonlinear.

Date: July 9, 2021.

2020 Mathematics Subject Classification. 65D15,41A46,65M60,49M99.

There is a huge variety of nonlinear approximation schemes, an extensive list goes far beyond the scope of this introduction. For reasons to be described below, we are interested in ridge functions for building nonlinear approximation schemes. More specific, we aim at constructing an approximation scheme consisting of a *linear part* and some *nonlinear enhancement* in terms of a ridge function update. Doing so, such an expansion consisting of a linear and a nonlinear part, we will be able to treat problems with fast and slow decay of $d_N(\mathcal{F})$ by the same algorithm. We need to collect some notation in order to motivate our choice.

1.1. **Linear/Ridge approximation.** We consider a given function $u : \Omega \to \mathbb{R}$, where $\Omega \subset \mathbb{R}^d$ is an open bounded domain and $u \in L_2(\Omega)$ is the minimal requirement, sometimes we need to assume more, e.g. $u \in C^{0,1}(\bar{\Omega})$, at least piecewise.[a] This function can be given either explicitly or implicitly as the (unknown) solution to a given problem. In order to formulate the approximation problem under consideration, let

$$X_N := \operatorname{span}(\Phi_N) \subset L_2(\Omega), \qquad \Phi_N := \{\varphi_1, ..., \varphi_N\}$$

be a given linear space of dimension $N \in \mathbb{N}$ with φ_i, $i = 1, ..., N$, being given functions (which do not need to be linearly independent). Next, we recall the notion of a ridge function and refer e.g. to [6, 17, 25] for overviews and details.

Definition 1.1. *Let $a \in \mathbb{R}^d$, $b \in \mathbb{R}$ and $v : \mathbb{R} \to \mathbb{R}$. Then, $w : \Omega \to \mathbb{R}$, $w(x) := v(a^\top x + b)$ is called* ridge function *with* profile v, direction a *and* offset b.

Remark 1.2. *In practical application, we shall assume $v \in L_2(\mathbb{R}) \cap C^{0,1}_{\mathrm{pw}}(\mathbb{R})$ for the profiles to be discussed below. We will denote its argument by $\xi \in \mathbb{R}$, i.e., $v(\xi)$.*

In addition to Φ_N, we assume that we are given a finite number $M \in \mathbb{N}$ of profiles

$$\mathcal{V}_M := \{v_1, ..., v_M\} \subset L_2(\Omega)$$

and consider the approximation problem

$$u(x) \approx \sum_{i=1}^{N} \alpha_i \varphi_i(x) + \sum_{j=1}^{M} c_j v_j(a_j^\top x + b_j) =: u_\delta(x), \qquad x \in \Omega. \tag{1.1}$$

A function u_δ of type (1.1) will be called *Linear/Ridge* expansion. Clearly, such a Linear/Ridge approximation depends on the choices of Φ_N and $\mathcal{V}_M$ as well as on the coefficients $\alpha_i, c_j \in \mathbb{R}$, the directions $a_j \in \mathbb{R}^d$ and the offsets $b_j \in \mathbb{R}$. However, in this paper we view Φ_N and $\mathcal{V}_M$ to be given (e.g. by some preprocessing training), so that we only consider the dependency of u_δ on the directions, offsets and coefficients. In order to shorten notation, we collect this dependency in one index δ. The main topic of this paper is thus to investigate appropriate choices of coefficients, directions and offsets in (1.1) in order to determine a "good" approximation of a given function u.

1.2. **Motivation.** The main source of motivation for this paper comes from model order reduction of parameterized partial differential equations (PPDE) by means of the reduced basis method (RBM, [14, 15, 26]). In order to briefly review it, let $L(\mu)$ be a parameterized partial differential operator and $f(\mu)$ be some given right-hand side, where $\mu \in \mathcal{P} \subset \mathbb{R}^Q$ is some parameter. One seeks for the exact solution $u(\mu)$ of $L(\mu)u = f(\mu)$ – in a suitable weak sense. Typically, a suitable (i.e., sufficiently fine, we say "detailed") discretization is available. However, the

[a] $C^{0,1}(\bar{\Omega}) := \{v \in C(\bar{\Omega}) : \exists L > 0 : \|v(x) - v(y)\| \le L \|x - y\| \text{ for all } x, y \in \Omega\}$.

computation of a numerical detailed approximation $u^{\mathcal{N}}(\mu)$ (with $\mathcal{N} \gg 1$ degrees of freedom) is too costly at least for approximating $u(\mu)$ for several different values of μ ("multi-query" context) and/or extremely fast (e.g. in realtime and/or in an embedded system).

The RBM aims at constructing a linear subspace $H_N \subset H$ (if the PPDE is posed on H, e.g. the Sobolev space $H_0^1(\Omega)$) of dimension $\mathbb{N} \ni N \ll \mathcal{N}$, which is typically constructed in an offline training phase. Then, in the online realtime or multiquery environment, a reduced approximation $u_N(\mu)$ is computed very rapidly – as the (Petrov-Galerkin) projection of the detailed solution $u^{\mathcal{N}}(\mu)$ onto H_N. In this context, the above mentioned class reads

$$\mathcal{F} = \{u(\mu) : \mu \in \mathcal{P}\}$$

and this explains the particular relevance of the decay of $d_N(\mathcal{F})$ for model order reduction, e.g. [1, 2]. In particular, it is known that certain elliptic problems admit an exponential decay of $d_N(\mathcal{F})$, [5], whereas parameterized transport and wave equations show poor decay, [13, 22], which motivates the construction of nonlinear model reduction (approximation) schemes in particular for such types of problems, [3, 4, 11, 20, 21, 27], just to mention a few.

We are suggesting a nonlinear update in terms of ridge functions due to several reasons. First of all, ridge functions have a simple structure and turn out the be particularly suitable for parameterized transport and wave-type problems as we shall see below. Second, there is a rich literature for approximation theory with ridge functions, see, e.g. [6, 9, 17, 25]. Finally, ridge functions are closely related to neural networks, which just recently have been investigated for nonlinear model reduction, see e.g. [10, 12, 19].

1.3. **Outline.** The remainder of this paper is organized as follows. In Section 2, we collect all required preliminaries and introduce the arising optimization problem. Section 3 is devoted to the association of the desired quantities (directions and offsets) with particles, which is the basis for the particle grid algorithm which we introduce in Section 4. Some results of our various numerical experiments are presented in Section 5. We finish with an outlook in Section 6.

2. Preliminaries

In order to quantify what has to be understood by 'good' approximation, define the set

$$U_{N,M} := \left\{ u_\delta(x) = \sum_{i=1}^{N} \alpha_i \, \varphi_i(x) + \sum_{j=1}^{M} c_j \, v_j(a_j^\top x + b_j), \;\; \alpha_i, b_j, c_j \in \mathbb{R},\, a_j \in \mathbb{R}^d \right\}, \tag{2.1}$$

which is a nonlinear subset of $L_2(\Omega)$. We consider the error in the L_2-norm and are interested in the/a best approximation in this norm abbreviated by $\|\cdot\|_0 \equiv \|\cdot\|_{L_2(\Omega)}$. In order to make (at least an approximation to) a best approximation accessible, we collect all variables by setting $\boldsymbol{\alpha} := (\alpha_i)_{i=1,\dots,N} \in \mathbb{R}^N$, $\mathbf{a} := (a_j)_{j=1,\dots,M} \in \mathbb{R}^{dM}$, $\mathbf{b} := (b_j)_{j=1,\dots,M}$, $\mathbf{c} := (c_j)_{j=1,\dots,M} \in \mathbb{R}^M$ and

$$\delta := (\boldsymbol{\alpha}, \mathbf{a}, \mathbf{b}, \mathbf{c}) \in \mathbb{R}^{N+(d+2)M}, \tag{2.2}$$

write $u_\delta(x)$ as in (2.1) and consider the *cost function*

$$J_u : \mathbb{R}^{N+(d+2)M} \to \mathbb{R}_{\geq 0}, \qquad J_u(\delta) := \|u - u_\delta\|_0^2. \tag{2.3}$$

Then, $u_{\delta^*} = \arg\inf_{u_\delta \in U_{N,M}} \|u - u_\delta\|_0$, where $\delta^* = \arg\inf_{\delta \in \mathbb{R}^{N+(d+2)M}} J_u(\delta)$.

Before we continue let us detail an example which also indicates our particular interest in such approximations to be considered here.

Example 2.1 (Parametric linear transport problem)**.** *Consider the homogeneous linear transport equation on the real line with velocity $\mu > 0$, i.e., $\partial_t u(t,x) + \mu\,\partial_x u(t,x) = 0$ on $(0,T)\times\mathbb{R}$, $T > 0$ with initial condition $u(0,x) = u_0(x)$, $x \in \mathbb{R}$. In a model reduction framework, one would interpret μ as a parameter and would want to approximate the solution $u(t,x;\mu) = u_0(x-\mu t)$ either for many velocities μ and/or in realtime. This is typically done in terms of a linear combination of "snapshots" $\varphi_i := u(\cdot,\cdot;\mu^{(i)})$, $i = 1,...,N$, where the snapshot parameters $\mu^{(i)}$ are chosen in some offline training phase. However, it is known from* [22] *that the Kolmogorov N-width is at most $\mathcal{O}(N^{-1/2})$, which makes such a linear model reduction inefficient.*

However, choosing the profile $v_1 := u_0$ and setting $a_1 := (-\mu, 1)^\top$, $b_1 := 0$, $c_1 := 1$ yields $c_1 v_1(a_1^\top(t,x) + b_1) = u_0(x-\mu t)$, i.e., the exact solution. Hence, choosing $M = 1$, $\mathcal{V}_1 = \{v_1\}$, we get that $\inf_\delta J_u(\delta) = 0$ independent of the choice of Φ_N.

We stress the fact that we cannot assume in general that the functions $v_1,...,v_M$ are linearly independent, not even pairwise. There are even cases, where $v_i = v_j$ for some distinct indices $i \neq j$ as the following example shows.

Example 2.2 (Parametric wave equation)**.** *Consider the linear wave equation $\partial_{tt}^2 u - \mu^2\,\partial_{xx}^2 u = 0$ for $t > 0$ and $x \in \mathbb{R}$ with initial conditions $u(0) = u_0$ and $\dot{u}(0) = 0$. The solution is given by the famous d'Alembert formula as $u(t,x;\mu) = \frac{1}{2}(u_0(x-\mu t) + u_0(x+\mu t))$. Hence, choosing $v_1 = v_2 = u_0$, $c_1 = c_2 = \frac{1}{2}$, $b_1 = b_2 = 0$ as well as $a_1 = (-\mu, 1)^\top$ and $a_2 = (\mu, 1)^\top$ yields the exact solution. This is an example of two identical profiles causing an exact representation using different directions. Besides, also for $\dot{u}(0) \neq 0$, the wave equation is a sum of two, but then different, ridge functions.*

Also this problem is particularly interesting since it is known that projection-based (i.e., linear) model reduction techniques do not work in the sense that the decay of the Kolmogorov N-width is at most $\mathcal{O}(N^{-1/4})$, [13]*. However, d'Alembert's solution formula is a ridge function.*

These examples should motivate the consideration of the optimization problem

$$J_u(\delta) \to \min! \quad \delta \in \mathbb{R}^{N+(d+2)M} \tag{2.4}$$

for a given function $u \in L_2(\Omega)$.

Lemma 2.3. *Let $u \in L_2(\Omega)$. Then, there exists a minimizer δ^* of* (2.4) *for J_u defined in* (2.3)*.*

Proof. Since $L_2(\Omega)$ is a Hilbert space and $U_{N,M} \subset L_2(\Omega)$ is closed, the claim follows by standard arguments. □

Note, that $U_{N,M}$ is not necessarily convex and therefore the minimizer in Lemma 2.3 is not necessarily unique. We are now going to determine an approximation $u_\delta \in U_{N,M}$ to the given function u step by step.

Remark 2.4. *In practice we might just have access to the function u indirectly by a quantity like the residuum of a PDE. In this case, one would not use the L_2-error as cost function J_u but rather the residuum (if computable) or an appropriate error estimator.*

2.1. **Given directions and offsets.** As a first step, let us assume that the directions $\mathbf{a}$ and offsets $\mathbf{b}$ according to the profiles $\mathcal{V}_M$ would be given. Then, the optimal coefficients $\boldsymbol{\alpha}$ and $\mathbf{c}$ for $\mathbf{a}$, $\mathbf{b}$ are easily determined as we shall see next.

Lemma 2.5. *Let $u \in L_2(\Omega)$ and Φ_N, $\mathcal{V}_M$, $\mathbf{a} = (a_j)_{j=1,\dots,M} \in \mathbb{R}^{dM}$ as well as $\mathbf{b} = (b_j)_{j=1,\dots,M}$ be given. Define the matrices $\mathbf{A}_{i,i'} := (\varphi_i, \varphi_{i'})_0$, $i, i' = 1, \dots, N$ and $\mathbf{C}_{j,j'} = (\mathbf{C}(\mathbf{a},\mathbf{b}))_{j,j'} := (v_j(a_j^\top \cdot + b_j), v_{j'}(a_{j'}^\top \cdot + b_{j'}))_0$ as well as $\mathbf{B}_{i,j} = (\mathbf{B}(\mathbf{a},\mathbf{b}))_{i,j} := (\varphi_i, v_j(a_j^\top \cdot + b_j))_0$, $i, i' = 1, \dots, N$, $j, j' = 1, \dots, M$. Then, the optimal coefficients $\boldsymbol{\alpha}^* = \boldsymbol{\alpha}^*(u,\mathbf{a},\mathbf{b}) = (\alpha_1^*, \dots, \alpha_N^*)^\top$ and $\mathbf{c}^* = \mathbf{c}^*(u,\mathbf{a},\mathbf{b}) = (c_1^*, \dots, c_M^*)^\top$ are given as the solution of the linear system of equations*

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ \mathbf{c} \end{pmatrix} = \begin{pmatrix} \mathbf{f}(u) \\ \mathbf{g}(u) \end{pmatrix}, \tag{2.5}$$

where the right-hand side is given by $\mathbf{f}(u)_i := (u, \varphi_i)_0$, $i = 1, \dots, N$, and $\mathbf{g}(u,\mathbf{a},\mathbf{b})_j := (u, v_j(a_j^\top \cdot + b_j))_0$, $j = 1, \dots, M$.

Proof. Let $\mathbf{a}$, $\mathbf{b}$ be fixed and denote $\tilde{\delta} := (\boldsymbol{\alpha}, \mathbf{c})$ as well as $\delta = (\boldsymbol{\alpha}, \mathbf{a}, \mathbf{b}, \mathbf{c})$ as in (2.2). We consider the reduced cost function $\tilde{J}_u(\tilde{\delta}) := J_u(\delta)$ as a function of $\tilde{\delta}$ only. Then, $\tilde{J}_u(\tilde{\delta}) = \|u - u_{\tilde{\delta}}\|_0^2 = \|u\|_0^2 - 2(u, u_{\tilde{\delta}})_0 + \|u_{\tilde{\delta}}\|_0^2$ and

$$\begin{aligned}
(u, u_{\tilde{\delta}})_0 &= \sum_{i=1}^N \alpha_i \, (u, \varphi_i)_0 + \sum_{j=1}^M c_j \, (u, v_j(a_j^\top \cdot + b_j))_0 = \mathbf{f}(u)^\top \boldsymbol{\alpha} + \mathbf{g}(u)^\top \mathbf{c}, \\
\|u_{\tilde{\delta}}\|_0^2 &= \sum_{i,i'=1}^N \alpha_i \, \alpha_{i'} (\varphi_i, \varphi_{i'})_0 + \sum_{j,j'=1}^M c_j \, c_{j'} \, (v_j(a_j^\top \cdot + b_j), v_{j'}(a_{j'}^\top \cdot + b_j))_0 \\
&\quad + 2 \sum_{i=1}^N \sum_{j=1}^M \alpha_i \, c_j (\varphi_i, v_j(a_j^\top \cdot + b_j))_0 = \boldsymbol{\alpha}^\top \mathbf{A} \boldsymbol{\alpha} + \mathbf{c}^\top \mathbf{C} \mathbf{c} + 2 \boldsymbol{\alpha}^\top \mathbf{B} \mathbf{c},
\end{aligned}$$

so that $\tilde{J}_u(\tilde{\delta}) = \|u\|_0^2 - 2\,\mathbf{f}(u)^\top \boldsymbol{\alpha} - 2\,\mathbf{g}(u)^\top \mathbf{c} + \boldsymbol{\alpha}^\top \mathbf{A} \boldsymbol{\alpha} + \mathbf{c}^\top \mathbf{C} \mathbf{c} + 2\,\boldsymbol{\alpha}^\top \mathbf{B} \mathbf{c}$. Hence, since $\mathbf{A}$ and $\mathbf{C}$ are symmetric,

$$\begin{aligned}
\nabla \tilde{J}_u(\tilde{\delta}) &= \begin{pmatrix} \partial_{\boldsymbol{\alpha}_N} J_u(\delta) \\ \partial_{\mathbf{c}_M} J_u(\delta) \end{pmatrix} = \begin{pmatrix} -2\,\mathbf{f}(u) + \mathbf{A}\boldsymbol{\alpha} + \mathbf{A}^\top \boldsymbol{\alpha} + 2\,\mathbf{B}\mathbf{c} \\ -2\,\mathbf{g}(u) + \mathbf{C}\mathbf{c} + \mathbf{C}^\top \mathbf{c} + 2\,\mathbf{B}^\top \boldsymbol{\alpha} \end{pmatrix} \\
&= 2 \begin{pmatrix} -\mathbf{f}(u) + \mathbf{A}\boldsymbol{\alpha} + \mathbf{B}\mathbf{c} \\ -\mathbf{g}(u) + \mathbf{C}\mathbf{c} + \mathbf{B}^\top \boldsymbol{\alpha} \end{pmatrix} = -2 \begin{pmatrix} \mathbf{f}(u) \\ \mathbf{g}(u) \end{pmatrix} + 2 \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ \mathbf{c} \end{pmatrix},
\end{aligned}$$

i.e., $\nabla \tilde{J}_u(\tilde{\delta}) = 0$ if and only if (2.5). Finally, $\nabla^2 \tilde{J}_u(\tilde{\delta}) = 2 \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{pmatrix}$ is symmetric and since for $x \in \mathbb{R}^{M+N}$

$$x^\top \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{pmatrix} x = \left\| \sum_{i=1}^N x_i \, \varphi_i + \sum_{j=1+N}^M x_j \, v_j(a_j^\top \cdot + b_j) \right\|_0^2 \geq 0,$$

$\nabla^2 \tilde{J}_u(\tilde{\delta})$ is also positive semi-definite, which concludes the proof. □

This result shows that the coefficients are determined once directions and offsets are given. Setting $\delta(u,\mathbf{a},\mathbf{b}) := (\boldsymbol{\alpha}^*(u,\mathbf{a},\mathbf{b}), \mathbf{a}, \mathbf{b}, \mathbf{c}^*(u,\mathbf{a},\mathbf{b}))$, we consider the reduced cost function depending solely on directions and offsets

$$\hat{J}_u : \mathbb{R}^{(d+1)M} \to \mathbb{R}_{\geq 0}, \qquad \hat{J}_u(\mathbf{a},\mathbf{b}) := \|u - u_{\delta(u,\mathbf{a},\mathbf{b})}\|_0^2 \tag{2.6}$$

along with the reduced optimization problem

$$\hat{J}_u(\hat{\delta}) \to \min! \qquad \hat{\delta} = (\mathbf{a}, \mathbf{b}) \in \mathbb{R}^{(d+1)M}. \tag{2.7}$$

Hence, we are left with finding the directions $\mathbf{a}$ and the offsets $\mathbf{b}$.

3. Directions, offsets and particles

Since the determination of directions and offsets amounts to solving a complex optimization problem, we aim at using a very well-known heuristic method, the particle swarm algorithm, see e.g. [16, 23]. In order to reduce computational complexity, we arrange our "particles" (which are associated to the collection of all directions $a_j \in \mathbb{R}^d$ and offsets $b_j \in \mathbb{R}$, $j = 1, ..., M$) in a dynamic grid. Hence, we call the arising scheme *particle grid* method. For each profile, we collect the direction and the offset in one vector $d_j := (a_j, b_j) \in \mathbb{R}^{d+1}$. These vectors are then associated to some component $p_j \in (-1,1)^D =: \mathbb{S}_D$. The vector $\mathbf{d} = (d_j)_{j=1,...,M} \in \mathbb{R}^{DM}$ of all directions and offsets is then associated to one *particle*. The dimension D is at most $d+1$, but can also be smaller if the problem at hand fixes some components of d_j, see Example 3.1 below.

Example 3.1. *Consider the linear transport and wave equation in one space-dimension from Example 2.1 and 2.2, respectively. The underlying domain is $\Omega = \mathbb{R}^+ \times \mathbb{R}$ and the variables read $(t,x) \in \Omega$ indicating time and space. Hence, in both cases, any direction takes the form $a = (a_t, a_x)$ and any ridge function reads $v(a_t\, t + a_x\, x + b)$ for some offset $b \in \mathbb{R}$.*

The ridge functions should be consistent with the initial condition $u(0,x) = u_0(x)$, which implies that $v := u_0$ and $a_x := 1$ as well as $b := 0$. This shows that we do not need a full vector $d = (a_t, a_x, b)$ of dimension $d+1 = 3$, but that a single parameter suffices to represent each combination of direction and offset with a particle.

In fact, in both cases, profiles take the form $v(x \pm \mu t)$, which means that we choose $a = (\pm\mu, 1)^\top$ and $b = 0$ with $\mu \in \mathbb{R}$.

The association of particles to all vectors d_j, $j = 1, ..., M$ (i.e., for all M profiles), is done by constructing an appropriate mapping

$$\pi : \mathbb{S}_D \to \mathbb{R}^{d+1}, \qquad \pi(p_j) =: (a_j, b_j)^\top.$$

Of course, on can construct several such mappings.

Example 3.2. *For $D = d+1$, we frequently used the smooth transformation $\pi : (-1,1)^{d+1} \to \mathbb{R}^{d+1}$ defined as $\pi_i(s) := \tan(\frac{\pi}{2} s)$ for each component $i = 1, ..., d+1$.*

Example 3.3 (Example 3.1 continued)**.** *In order to associate the real-valued parameter μ to a particle $p \in (-1,1)$, we can define $\pi : \mathbb{S}_1 \to \mathbb{R}^3$ by $p \mapsto (\tan(\frac{\pi}{2}p), 1, 0)^\top$, i.e., $\mu = \tan(\frac{\pi}{2}p)$.*

This setting now allows us to reformulate the reduced optimization problem (2.7) in terms of the particles, i.e., $\hat{J}_u(\mathbf{p}) \to \min!$ for $\mathbf{p} = (p_1, ..., p_M) \in \mathbb{S}_D^M$, where

$$\hat{J}_u : \mathbb{S}_D^M \to \mathbb{R}_{\geq 0}, \quad \hat{J}_u(\mathbf{p}) := \check{J}_u(\tau(p_1), ..., \tau(p_M)), \tag{3.1}$$

where the arguments are to be understood in the following manner:

$$\mathbb{R}^{(d+1)M} \ni \hat{\delta} = \begin{pmatrix} a_1 & a_2 & \cdots & a_M \\ b_1 & b_2 & \cdots & b_M \end{pmatrix} = (d_1, ..., d_M) = (\tau(p_1), ..., \tau(p_M)).$$

We call such a $\mathbf{p}$ *particle*.

Example 3.4. *In order to illustrate the challenges for solving the reduced optimization problem for* (3.1)*, let us consider three examples, where the function u to be approximated is given as a sum of ridge functions (i.e., which can even be represented exactly). We consider the two-dimensional case $\Omega = (0,1)^2$ and $M = 2$ profiles. The data is collected in the following table.*

Case	$u(x_1, x_2)$	$v_1(\xi)$	$v_2(\xi)$
1	$1.6\cos(10x_1 + \frac{10}{3}x_2) + 0.8\cos(10x_1 - 5x_2)$	$\cos(10\xi)$	$\cos(10\xi)$
2	$\cos(x_1 + 2x_2) + \cos(x_1 - 0.5\,x_2)$	$\cos(\xi)$	$\cos(\xi)$
3	$5\,\lvert x_1 - \frac{1}{2}x_2 - \frac{1}{2}\rvert + 0.6\,(x_1 + \frac{1}{3}x_2)^2$	$\lvert\xi - \frac{1}{2}\rvert$	ξ^2

Given v_1 and v_2, setting $b_1 = b_2 = 0$, $(a_1)_1 = (a_2)_1 = 1$ we scatter $\mathbb{S}_1^2$ by selecting particles $p = (p_1, p_2) \in (-1,1)^2$ for the remaining two unknowns $(a_1)_2$ and $(a_2)_2$. Each such particle defines a direction for which we define u_δ and compute the value of the cost function, i.e., the error $\|u - u_\delta\|_0$. The results are depicted in Figure 1.

The functions u are shown in the top row. The reduced cost function $\hat{J}_u$ as a function of $p \in (-1,1)^2$ is visualized in the second row, scattered on a grid of 129^2 points. Solving for the two directions and offsets would thus amount finding a global minimum of the cost functions shown in the second row. As we see, we may face multiple local minima, even multiple global minima (in particular in cases 1 and 2, where $v_1 = v_2$, so that we see symmetry), steep gradients and several highly localized phenomena.

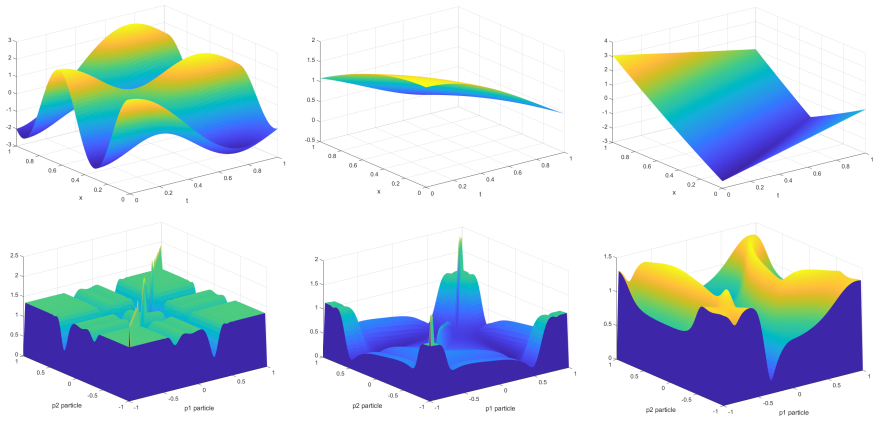


Figure 1. Cases 1-3 (from left to right): function to be approximated (top row) and corresponding reduced cost functions (bottom row).

As we see from Example 3.4, we may face a truly complex optimization problem. In particular, we cannot hope to use gradient-based optimization techniques, at least not in a straightforward manner.

4. A particle grid algorithm

As already motivated earlier, we are now going to describe an algorithm which has shown good performance in determining at least a good approximation to a global minimum of $\hat{J}_u$ in (3.1). Recall, that we are facing an optimization problem

in $\mathbb{S}_D^M \cong \mathbb{S}_M^D \cong \mathbb{S}_{DM} = (-1,1)^{DM}$. Hence, we define the optimization dimension as $P := DM$.

The algorithm produces a sequence of *particle grids*

$$\mathbf{P}^{(0)} \to \mathbf{P}^{(1)} \to \dots \to \mathbf{P}^{(k)} \to \mathbf{P}^{(k+1)} \to \dots,$$

where each grid (i.e., a swarm in form of a grid) $\mathbf{P}^{(k)}$ consists of m_{par} particles in $\mathbb{S}^P$. We choose n_{par} nodes in each dimension, i.e.,

$$m_{\text{par}} = n_{\text{par}}^P \quad \text{for} \quad n_{\text{par}} \in \mathbb{N},$$

particles in $\mathbb{S}^P = (-1,1)^P$. Then, we initialize the first particle swarm $\mathbf{P}^{(0)}$ by taking the tensor product, yielding a regular grid. Each particle has uniquely defined next neighbors in each diagonal direction. This next neighbor relation does not change in the course of the iteration, as we will see in Lemma 4.1. This means that each swarm is a *grid* whose internal geometry does not change even if the position of each particle may vary. We may associate each particle grid $\mathbf{P}^{(k)}$ with a tensor of dimension P (e.g., a matrix for $P = 2$).

If $\mathbf{P}^{(k)} = \{\mathbf{p}_{\mathbf{i}}^{(k)} \in (-1,1)^P : \mathbf{i} = (i_1, \dots, i_P) \in \{1, \dots, n_{\text{par}}\}^P\}$, and each particle takes the form $\mathbf{P}^{(k)} \ni \mathbf{p}_{\mathbf{i}}^{(k)} = ((\mathbf{p}_{\mathbf{i}}^{(k)})_1, \dots, (\mathbf{p}_{\mathbf{i}}^{(k)})_P)^\top \in (-1,1)^P$, the algorithm can be described as follows: Choose $\delta \in (0, \frac{1}{2})$. Then, for each particle $\mathbf{p}_{\mathbf{i}}^{(k)} \in \mathbf{P}^{(k)}$, we consider the value of the particle and of the at most $3^P - 1$ surrounding particles $\mathbf{p}_{\mathbf{j}}^{(k)}$ with $\mathbf{j} = (j_1, \dots, j_P)$ where $|i_s - j_s| \le 1$ for all $1 \le s \le P$. We collect the indices of these particles in a set $\mathcal{I}(\mathbf{i}) := \{\mathbf{j} = (j_1, \dots, j_P) \in \{1, \dots, n_p\}^P : |i_s - j_s| \le 1,\ 1 \le s \le P\}$ and set

$$\mathbf{q}_{\mathbf{i}}^{(k+1)} := \arg\min_{\mathbf{j} \in \mathcal{I}(\mathbf{i})} \hat{J}_u(\mathbf{p}_{\mathbf{j}}^{(k)}), \qquad |\mathcal{I}(\mathbf{i})| \le 3^P - 1. \tag{4.1}$$

Subsequently we get the next particle by the step:

$$\mathbf{p}_{\mathbf{i}}^{(k+1)} := (1-\delta)\mathbf{p}_{\mathbf{i}}^{(k)} + \delta \mathbf{q}_{\mathbf{i}}^{(k+1)}. \tag{4.2}$$

For the points on the boundary of the grid we need to slightly adapt this procedure. Technically, we view particles at opposite sides of the boundary as being neighbors, which means that, for example in one dimension the particle on the most left is considered as the neighbor of the particle on the most right. For dimension $1 \le s \le P$, a particle $\mathbf{p}_{\mathbf{j}}^{(k)}$ with $\mathbf{j} = (j_1, \dots, j_{s-1}, 1, j_{s+1}, \dots, j_P)$ is a neighbor of $\mathbf{p}_{\mathbf{i}}^{(k)}$ with $\mathbf{i} = (j_1, \dots, j_{s-1}, n_{\text{par}}, j_{s+1}, \dots, j_P)$. This means that we are sticking the boundary together, so that we get a grid where in each dimension each particle has neighbors to the left and to the right, similar to a torus.

This concludes the description of one iteration $\mathbf{P}^{(k)} \to \mathbf{P}^{(k+1)}$, which is terminated after a predefined number $K \in \mathbb{N}$ of iterations with the output

$$\mathbf{p}_{\text{app}} := \arg\min_{\mathbf{i} \in \{1, \dots, n_p\}^P} \hat{J}_u(\mathbf{p}_{\mathbf{i}}^{(K)}). \tag{4.3}$$

which is used as an approximation for some minimizer $\mathbf{p}^*$. Here, we often choose $\delta = \frac{1}{3}$, which turned out to be a reasonable choice as it always ensures a positive distance of neighboring particles. We summarize the method in Algorithm 1.

Algorithm 1 Particle grid algorithm

Input: Number of particles $m_{\text{par}} = n_{\text{par}}^P$, maximal number of iterations K, parameter $\delta \in (0, \frac{1}{2})$

1: Initialize $\mathbf{P}^{(0)} = \{\mathbf{p}_{\mathbf{i}}^{(0)} : \mathbf{i} \in \{1, ..., n_{\text{par}}\}^P\}$ equidistantly
2: **for** $k = 0$ **to** $K - 1$ **do**
3: **for** all indices $\mathbf{i} \in \{1, ..., n_{\text{par}}\}^P$ **do**
4: Compute $\mathbf{q}_{\mathbf{i}}^{(k+1)}$ as in (4.1)
5: Calculate $\mathbf{p}_{\mathbf{i}}^{(k+1)}$ by (4.2); set $\mathbf{P}^{(k+1)} = \{\mathbf{p}_{\mathbf{i}}^{(k+1)} : \mathbf{i} \in \{1, ..., n_{\text{par}}\}^P\}$
6: Compute $\hat{J}_u(\mathbf{p}_{\mathbf{i}}^{(k+1)})$
7: **end for**
8: Get $\mathbf{p}_{\text{app}}$ as in formula (4.3)
9: **end for**

Output: Approximation particle $\mathbf{p}_{\text{app}}$

Figure 2 shows 6 stages of Algorithm 1 for one specific example with $P = DM = 2$ and $n_{\text{par}} = 10$. We start with a regular grid on top left and indicate how the algorithm moves the particles by showing the states for iterations $k = 1$, $k = 5$, $k = 25$, $k = 56$ and $k = 90$. As we see, not all particles are concentrated in one point even for this case having a single global minimum. The reason is that sticking the opposite boundaries together prevents a concentration. However, most points are very close to the global minimum or on lines parallel to the coordinate axes through the point of global minimum.

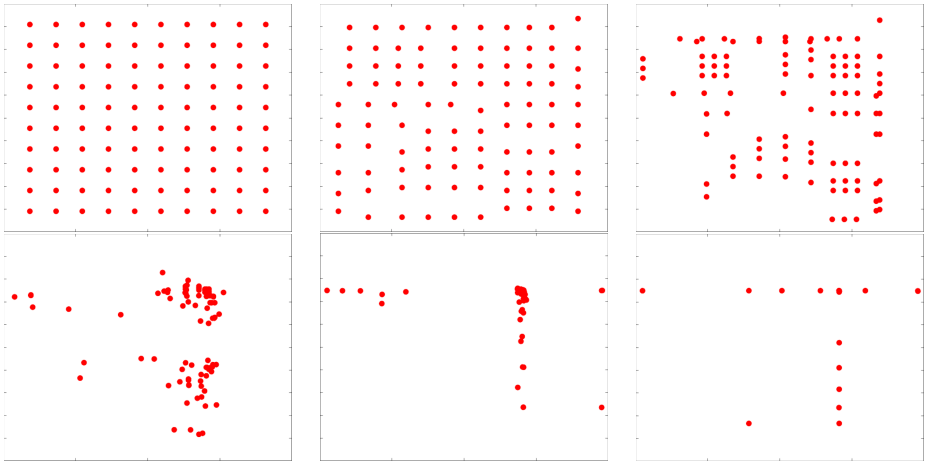


Figure 2. Particle grid iterations for $P = DM = 2$, $n_{\text{par}} = 10$: Iterations $k = 0$, $k = 1$, $k = 5$, $k = 25$, $k = 56$, $k = 90$.

The following observation is then immediate.

Lemma 4.1. *The next neighbor relation does not change in the course of the iteration, i.e., the index set $\mathcal{I}(\mathbf{i})$ is independent of the iteration k.* □

4.1. **Complexity.** Let K be the number of iterations and m_{par} the number of particles. In order to compute $\hat{J}_u$, we sample $\Omega \subset \mathbb{R}^d$ by $n \in \mathbb{N}$ quadrature points for computing the L_2-norm. Hence, we need to compute point values of u and u_δ at n^d points. For each quadrature point, the evaluation of u_δ requires to evaluate φ_i, $i = 1, ..., N$ and $v_j(a_j^\top \cdot + b_j)$, $j = 1, ..., M$, which is a total of $\text{Cost}(\hat{J}_u) := N \cdot n^d + M \cdot (d+1) \cdot n^d$ operations for one evaluation of J_u and $m_{\text{par}} \cdot \text{Cost}(\hat{J}_u)$ evaluations for one particle grid $\mathbf{P}^{(k)}$. For each iteration, we need to evaluate $\hat{J}_u$ for all grid points and per grid point, we have at most $3^P - 1$ comparisons. In total, Algorithm 1 thus requires in the order of

$$K \cdot n_{\text{par}}^P \cdot \left[n^d \cdot (N + M \cdot (d+1)) + 3^P - 1 \right]$$

operations, where $P = DM$ and $D \leq d+1$. Hence, the algorithm gets to its limits for large particle dimension D and large numbers M of profiles. The dimension N of the linear part only plays a minor role, especially as with sufficient memory the values of φ_i can be computed once and be stored.

4.2. **Parallelization.** The particle grid algorithm has the advantage that it can easily be parallelized. In fact, in a shared memory environment, the current grid $\mathbf{P}^{(k)}$ and any given particle $\mathbf{p}_\mathbf{i}^{(k)}$ is needed for determining the position $\mathbf{p}_\mathbf{i}^{(k+1)}$ in the next iteration. Hence, all such computations can be performed in parallel without further communication, which leads to linear speedup w.r.t. the numbers of processors. For distributed memory one would pass the positions of the neighbors of $\mathbf{p}_\mathbf{i}^{(k)}$ to the processor handling this particle.

5. Numerical experiments

In this section, we present results of some of our numerical examples. The main focus is the question how well a Linear/Ridge expansion is able to approximate certain functions and also the quantitative performance of the presented particle grid algorithm.

We start by applying our particle grid algorithm for determining a Linear/Ridge approximation of type (1.1), i.e., we seek for an approximation in $U_{N,M}$ consisting of the sum of a linear combination of functions Φ_N and profiles $\mathcal{V}_M$ for selected choices of such sets Φ_N and $\mathcal{V}_M$.

5.1. **Approximation properties.** First, we choose functions u which can be written exactly in the form (1.1), i.e., $u \in U_{N,M}$ and approximate the coefficients $\mathbf{y}$ in (2.2) by our particle grid algorithm. Doing so, we can monitor the error $\|u - u_\mathbf{y}\|_0$.

5.1.1. *Polynomials and Wavelets.* We start by a problem in two dimensions, $(t, x) \in \Omega = [0,1] \times [-1,1]$ which might be interpreted as time and space. It seems to be a straightforward choice to use $\text{span}(\Phi_N) = \mathcal{P}_r(\Omega)$, i.e., the space of algebraic polynomials of degree at most $r \in \mathbb{N}$. Here we choose $r = 2$, so that $N = 6$ and for simplicity we use the monomial basis, i.e.,

$$\varphi_1(t,x) = 1,\ \varphi_2(t,x) = t,\ \varphi_3(t,x) = x,\ \varphi_4(t,x) = t^2,\ \varphi_5(t,x) = tx,\ \varphi_6(t,x) = x^2.$$

Of course, we could use a more stable basis Φ_6 (e.g. orthonormal polynomials), but we are also interested to see how the algorithm can cope with ill-conditioned sets Φ_N, which are possibly even allowed to be linearly dependent.

Concerning the profiles, we choose wavelets as it is known that dilates and translates of wavelets yield frames or bases of $L_2(\mathbb{R}^2)$. That makes them good candidates for profiles. Specifically, we take $M = 4$ and

- v_1 as Mexican Hat,
- v_2 as the Haar wavelet,
- v_3 as Morlet wavelet,
- and v_4 to be the Hockeystick, which is also known as the ReLU activation function in neural networks,

see Figure 3b. The function u to be approximated is a Linear/Ridge expansion with $\alpha_i = 1$, $i = 1, ..., 6$, $b_j = 0$, $c_j = 1$, $j = 1, ..., 4$ and $a_1 = (1, 1)^\top$, $a_2 = (1, \sqrt{2} - 1)^\top$, $a_3 = (1, -1)^\top$ and $a_4 = (1, \sqrt{2} + 1)^\top$. The arising function is displayed in Figure 3a. As we see, the shape of u is rather complex. As the required discretization for performing computations, we use $h_t = \frac{1}{128}$, $h_x = \frac{1}{64}$, so that we have 129 grid points for both variables. For Algorithm 1 we choose as above $\delta = 1/3$ and $m_{\text{par}} = 6^4 = 1296$ particles. In Figure 3d, we monitor the L_2-error over the number of iterations. We obtain monotone convergence, without rate of course. For example, in order to reach an error smaller than 10^{-4} we need about 250 Iterations. After about 1000 iterations, the L_2-error is machine accuracy, i.e., $9.91 \cdot 10^{-15}$. The approximation u_δ is shown in Figure 3c.

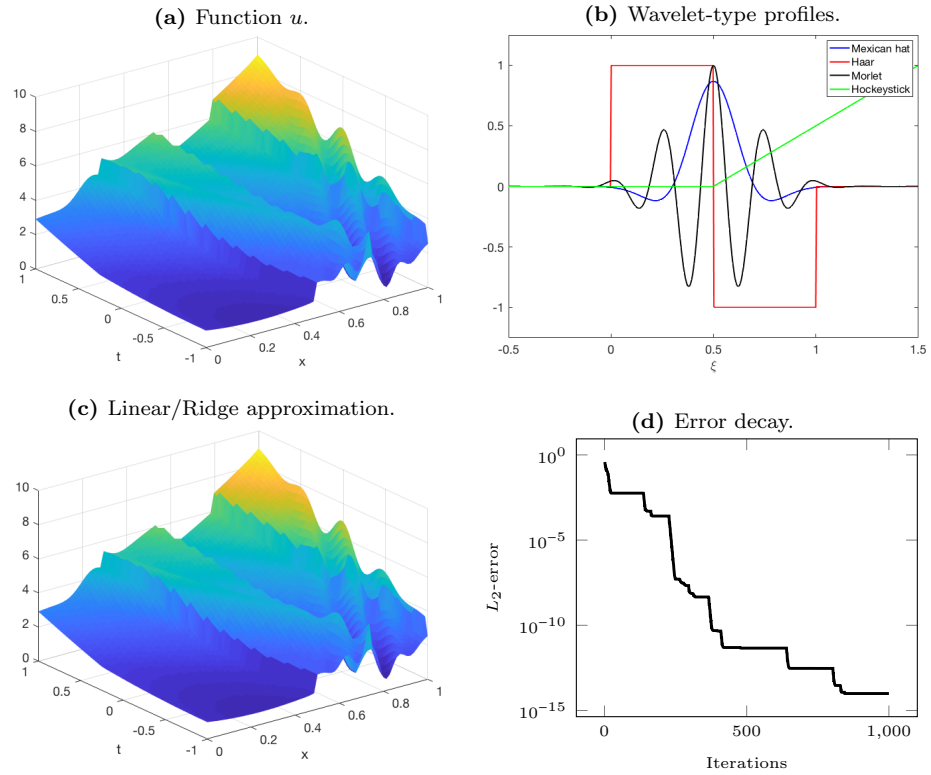
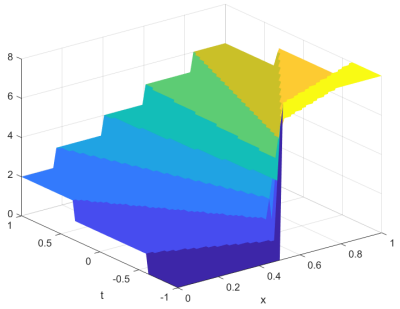


Figure 3. Approximation of function given as sum of quadratic polynomials ($N = 6$) and $M = 4$ wavelets of different type.

5.1.2. *Discontinuous staircase function.* Even though the shape of the function in our first example is complex, it is a smooth function[b]. Hence, we are now going to consider a function which consists of eight jumps along straight lines which are rotated in order to exclude a tensor product approximation. We use again $\Omega = (0,1) \times (-1,1)$. The profiles are chosen as the jump function displayed in Figure 4a top right. In order to investigate the role of non-linearly independence of the profiles, we take $v_1 = \cdots = v_8 = \mathbb{1}_{\mathbb{R}^+}$, i.e., $M = 8$ identical profiles. We forego the linear part here, i.e., $N = 0$. The function u to be approximated takes the form (1.1) with $N = 0$ and $M = 8$ choosing $b_j = 0$, $c_j = 1$, $j = 1, ..., 8$ and directions according to the angles $\frac{\pi}{9}$, so that we obtain a staircase-like function as displayed in Figure 4a bottom left.

For the discretization, we choose as above 129 grid points in both coordinate directions. For the particle grid algorithm, we use $\delta = 1/3$, $m_{\text{par}} = 3^8 = 6561$ and $m_{\text{par}} = 4^8 = 65563$ particles, respectively, where we note that the algorithm did not converge for 2^8 particles. The error decay is shown in Figure 4b. The reason why we show results for $m_{\text{par}} = 3^8$ and $m_{\text{par}} = 4^8$ is the fact that we observe a stagnation of the error for $m_{\text{par}} = 3^8$ after 50 iterations, whereas $m_{\text{par}} = 4^8$ yields machine accuracy after 49 iterations. This shows that one might be forced to use a large number of particles in order to reach high accuracies. After 50 iterations, we achieved an L_2-error of 0. In Figure 4a, we show the staircase-type step function that is also the computed approximation after 50 iterations.

(a) Staircase-type step function.



(b) Error decay.

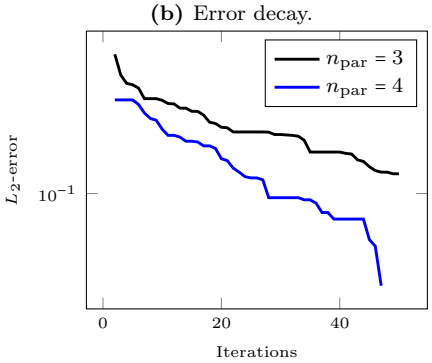


Figure 4. Approximation of a given staircase-type step function.

5.2. **Non-exact approximation functions.** So far, we used the algorithm to approximate functions that can exactly be represented in terms of the chosen families Φ_N and $\mathcal{V}_M$. This shows how fast (or slow) the algorithm is able to find the function in terms of directions and offsets. Now, we are going to consider functions u that cannot be represented exactly by u_δ, i.e., $\inf_\delta \|u - u_\delta\| > 0$. To this end, consider for $c \geq 1$ the function

$$u(t,x;\mu) := \sum_{k=1}^{\infty} \frac{1}{k!} \cos\left[2\pi\left(k\tfrac{\sqrt{c}}{10}\, t + x\right)\right], \qquad (t,x) \in \Omega := (0,1) \times (-1,1). \tag{5.1}$$

[b]Of course, the Haar wavelet is discontinuous, but a quite easy and single function to be approximated.

For the linear part of the approximation, we choose *snapshots* $\varphi_j(t,x) = u(t,x;j)$, for $j = 1,...,N$, and the profiles are chosen as $v_i := \cos(2\pi\cdot)$, $i = 1,...,M$ for $M \in \{0,...,4\}$. In Figure 5, we display the L_2-error for $u(t,x) = u(t,x;100)$ and different values of N and M. For the discretization, we choose as above 129 grid points in both coordinate directions and for the particle grid algorithm, we used $\delta = 1/3$, $m_{\text{par}} = 5^M$ particles. We do not obtain monotone convergence as N grows, which might be a stability issue. On the other hand, however, the error decreases for fixed N and increasing M in a monotonic manner.

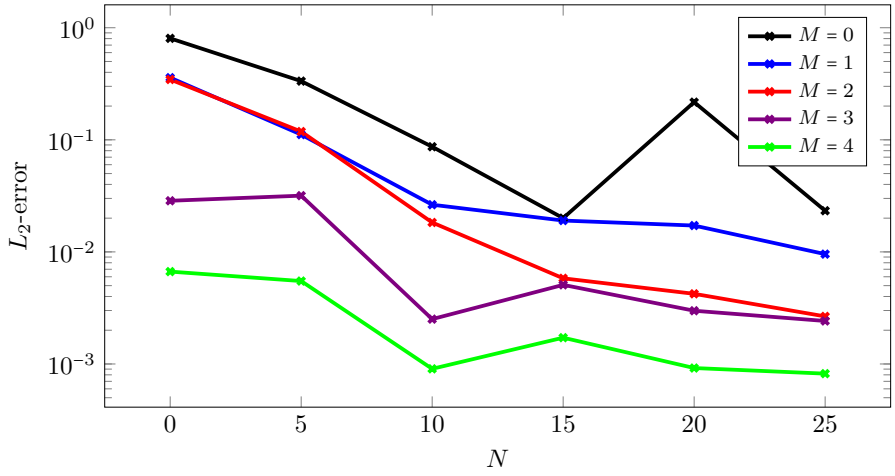


Figure 5. Approximation of an infinite series: L_2-error for different sizes of Φ_N and $\mathcal{V}_M$.

5.3. **Parametric Partial Differential Equations (PPDEs).** Next, we consider two PPDEs that depend on parameters $\mu \in \mathbb{R}^P$ and investigate the approximation of the solution $u(\mu)$ for various parameter values. The first problem is a stationary one, the second is instationary; both are defined in $\Omega = (0,1)^2$, which is interpreted as a time/space-domain for the wave equation. For both cases, we use $\delta = 1/3$ and $m_{\text{par}} = 11^2 = 121$ particles.

5.3.1. *Case 1: Thermal block.* The thermal block is a classical problem for model reduction, [14]. We consider the stationary case, which is a Poisson problem with piecewise constant coefficients, which serve as parameters. In fact, we decompose $\overline{\Omega}$ by $\overline{\Omega}_i := [0,1] \times [\frac{i-1}{4}, \frac{i}{4}]$, $i = 1,...,4$. The bilinear form of the variational formulation of the problem reads for $\mu = (\mu_1,...,\mu_4) \in \mathbb{R}^4$, $P = 4$, as follows

$$a(u,v;\mu) = \sum_{i=1}^{4} \mu_i \int_{\Omega_i} \nabla u(x)\, \nabla v(x)\, dx.$$

The right-hand side and boundary data can be retrieved from [14], and the formula for the exact solution can easily be seen to be for $x = (x_1, x_2) \in \Omega$ as follows

$$u(x;\mu) = \sum_{i=1}^{4} \frac{1}{\mu_i} \varphi_i(x), \quad \varphi_i(x) = \begin{cases} \frac{1}{4}, & 0 \le x_2 < \frac{i-1}{4}, \\ -x_2 + \frac{i}{4}, & x_2 \in [\frac{i-1}{4}, \frac{i}{4}], \\ 0, & \frac{i}{4} < x_2 \le 1, \end{cases}$$

see the left part of Figure 6. We consider different parameters $\mu \in [0.1, 10]^4$.

Due to the specific form of the solution, we use the linear approximation with $N = 4$ and choose $\Phi_4 := \{\varphi_1, \varphi_2, \varphi_3, \varphi_4\}$ with the functions defined above. We took $M = 2$ arbitrarily chosen profiles. The reason for this choice is as follows: Since the solution can be approximated quite accurately by the linear combination of the *snapshots* φ_i (which is due to the fact that such snapshots are known from the Reduced Basis Method to yield a very accurate approximation), the particle grid algorithm should automatically detect that no profiles are needed. As we can see from the results in Table 1 this is in fact the case – the algorithm finds the solution up to machine accuracy after only 1 iteration of the particle grid algorithm. We stress the fact that we tested much more parameter values and always observed this behavior.

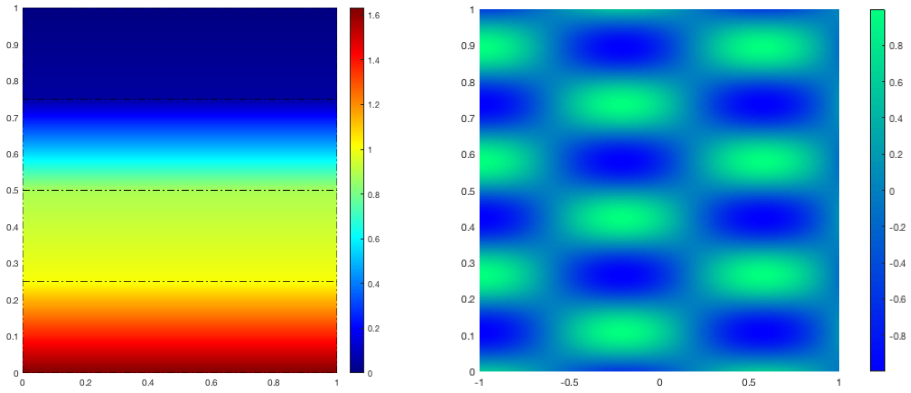


Figure 6. Exact solutions for the PDE examples 1 and 3. Left: 4×1 thermal block for $\mu = (0.4, 2, 0.3, 5)$. Right: Solution to wave equation for $\mu = 0.8$.

5.3.2. *Case 2: Linear transport equation.* Recalling Example 2.1, we consider the parametric linear transport equation with initial condition u_0, which we choose as a single profile, i.e., $M = 1$. For the linear part we use the same Φ_4 as in §5.3.1, i.e., the algorithm is supposed to set the corresponding coefficients α_i to zero.

5.3.3. *Case 3: The wave equation.* As in Example 2.2, we consider the univariate parametric wave equation $u_{tt}(t,x) - \mu^2 u_{xx}(t,x) = 0$ for $(t,x) \in \Omega$ along with initial conditions. We choose the initial data in such a manner that the solution reads

$$u(t,x;\mu) = 0.5 \sin(10x + 10\mu t) + 0.5 \sin(10x - 10\mu t),$$

which is a superposition of two ridge functions. Consequently, we set $M = 2$ and $\mathcal{V}_2 := \{v_1, v_2\} = \{\sin(10\cdot), \sin(10\cdot)\}$. Again, for the linear part we use Φ_4 as in §5.3.1.

The results are shown in Table 1. As expected, the hyperbolic wave equation is a harder problem than linear transport, which in turn is much harder problem than the thermal block. The numbers of the required iterations of the particle grid algorithm are significantly higher in order to reach a desired tolerance. On the other hand, we see that the algorithm is able to dismiss all the linear functions. The algorithm in fact converges and even reaches machine accuracy – at the expense of more iterations.

Case	PPDE	parameter μ	no. iterat. K	L_2-error
1	Thermal block	$(0.1, 10, 1, 0.6)$	1	$5.1019e-15$
1	Thermal block	$(10, 2, 0.1, 0.5)$	1	$1.4446e-14$
1	Thermal block	$(0.4, 2, 0.3, 5)$	1	$6.0861e-15$
2	Transport	1/4	12	$7.1348e-05$
2	Transport	1/4	66	$4.6205e-16$
2	Transport	1	19	$3.0119e-05$
2	Transport	1	70	$9.3829e-17$
2	Transport	4	24	$6.1985e-05$
2	Transport	4	67	0
3	Wave	1/4	20	$7.6682e-05$
3	Wave	1/4	83	$8.9850e-16$
3	Wave	1	21	$8.2400e-05$
3	Wave	1	86	$3.1765e-16$
3	Wave	4	28	$4.3012e-05$
3	Wave	4	83	$8.9850e-16$

Table 1. Parametric PDEs: Errors and iterations for both examples and different parameter values.

5.4. **Higher dimensions.** Next, we are considering a problem in higher dimensions, namely travelling plane waves (i.e., a wave function that is constant over any plane that is orthogonal to a fixed direction in space). The general form of a plane wave thus reads for a given normalized direction $n = (n_1, n_2, n_3)^\top \in \mathbb{R}^3$, $\|n\| = 1$ and some velocity c as follows

$$A(t, x) := v\big(n^\top x - c\ t\big), \quad v : \mathbb{R} \to \mathbb{R},$$

which is actually a ridge function. It is well-known that A can also be obtained as the solution of the 3d-wave equation $A_{tt} - c^2 \Delta A = 0$, which is the connection to our previous example. From a physical point of view, we shall assume that the plane waves emerge from two different light sources, i.e., for fixed t it takes the form

$$u(x) := \sum_{i=1}^{2} v_i\big(n_i^\top x - c\ t\big),$$

which we use as function to be approximated. We choose the specific profiles $v_1 := v_2 := \sin$, i.e., again, identical profiles. We fix time and velocity as $t = 1$, $c = 1$. Doing so, we get the directions $\mathbf{a} \in \mathbb{R}^6$ for $d = 3$, $M = 2$, i.e., $D = nM = 6$ as $a_j = n^j \in \mathbb{R}^3$, $j = 1, 2$ and $\mathbf{b} = (b_1, b_2)$, $b_1 = b_2 = -c$. We use the specific choices $n_1 := (-1/\sqrt{2}, 1/\sqrt{2}, 0)$ and $n_2 := (0, -1/\sqrt{2}, 1/\sqrt{2})$.

We are interested in the convergence history of the particle grid algorithm. To this end, we fix the number of particles $n_{\text{par}} = 4$ per direction, i.e., a total of $m_{\text{par}} = n_{\text{par}}^D = 4^6 = 4096$. The results are shown in Figure 7. We obtain quite fast convergence at the early stages which then slows down.

5.5. **Training directions for parametric PDEs.** Let us reconsider parameter-dependent PDEs as in §5.3 above. Within a model reduction context, a reduced

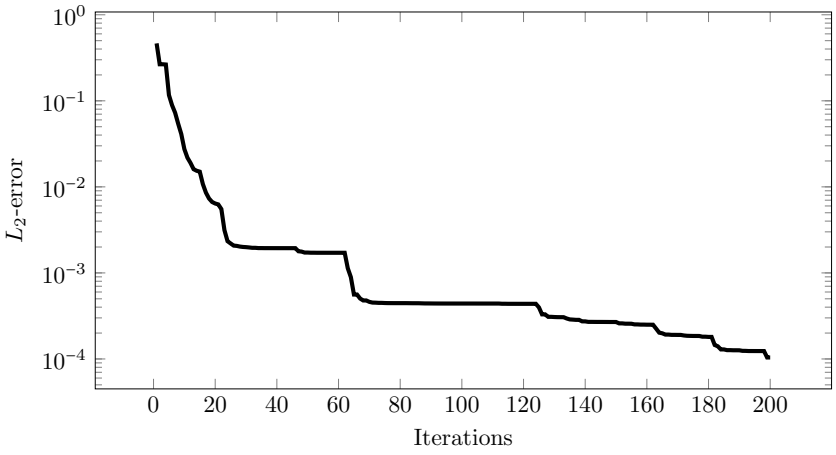


Figure 7. 3d planar wave. L_2-error over iterations for fixed number $n_{\mathrm{par}} = 69$ of particles. The minimal error is 1.040410^{-4}.

model is typically determined offline within a training phase. In our setting the sets Φ_N and $\mathcal{V}_M$ would thus be determined in an offline training.

In an online environment (typically suitable for multi-query or realtime situations), new parameters $\mu \in \mathbb{R}^P$ are given and an approximation is to be determined extremely fast – in online complexity. If we would determine the online approximation in terms of u_δ in (1.1), we would need to determine (optimal) directions that are parameter-dependent, i.e., $\mathbf{a}(\mu)$ and $\mathbf{b}(\mu)$, which might be computationally costly in the sense that many iterations of the particle grid algorithm would be needed, destroying online efficiency.

In order to speedup this procedure, we suggest to train the mapping $\mu \mapsto (\mathbf{a}(\mu), \mathbf{b}(\mu))$ offline as follows: For a given (or to be determined) set $\{\mu_1, ..., \mu_{n_{\mathrm{train}}}\}$ of training parameters, we determine (highly accurate) approximations $\mathbf{a}(\mu_i), \mathbf{b}(\mu_i)$, $i = 1, ..., n_{\mathrm{train}}$, offline by the particle grid algorithm. Then, we determine a interpolation $\mu \mapsto (\check{\mathbf{a}}(\mu), \check{\mathbf{b}}(\mu))$ of those offline data such that the interpolation error $\|(\mathbf{a}(\mu), \mathbf{b}(\mu)) - (\check{\mathbf{a}}(\mu), \check{\mathbf{b}}(\mu))\|$ is small, at least for a number of test samples μ. Of course, this interpolation error influences the overall online approximation error. We are going to investigate this influence.

To this end, we consider the solution $u(\cdot, \cdot\,; \mu)$ of the parametric linear transport problem $u_t + \mu^{-1/2} u_x = 0$, $u(0) = u_0$, since it is known that this problem also results in poor approximation rates using standard linear model reduction such as the Reduced Basis Method, [22]. In addition, the function $\mu \mapsto \mu^{-1/2}$ is much harder for the interpolation as a polynomial parameter-dependence. We choose 10 equidistant training parameters $\mu_i \in [0.1, 1]$. Since the offset is zero here, we only determined the corresponding directions $a(\mu_i)$, $i = 1, ..., 10$, which are real numbers here. The left graph in Figure 8 shows a cubic spline interpolation of the obtained data. We also display the exact curve $\mu \mapsto a(\mu)$ by determining high resolution approximations of optimal directions for 100 parameters μ by the particle grid algorithm. As we see, the error is almost negligible.

Next, we computed an online approximation for a new parameter μ as follows: (1) Evaluate the interpolation to retrieve $\check{a}_j(\mu)$, set $b_j = 0$, $N = 0$; (2) compute the

coefficients c_j and obtain an approximation $\check{u}_\delta$ in (1.1). This is to be compared with the function u_δ using the exact direction $a_j(\mu)$. On the right in Figure 8, we display these errors $\|\check{u}_\delta(\cdot,\cdot;\mu) - u_\delta(\cdot,\cdot;\mu)\|_0$ for different initial conditions u_0 which also serve as the single profile. We used the same knots for the interpolation and see that the quantitative errors depend on the initial condition or more precisely on the derivative of the profile. However, please note the range of the vertical axis, which indicates that all errors are in fact in a comparable range.

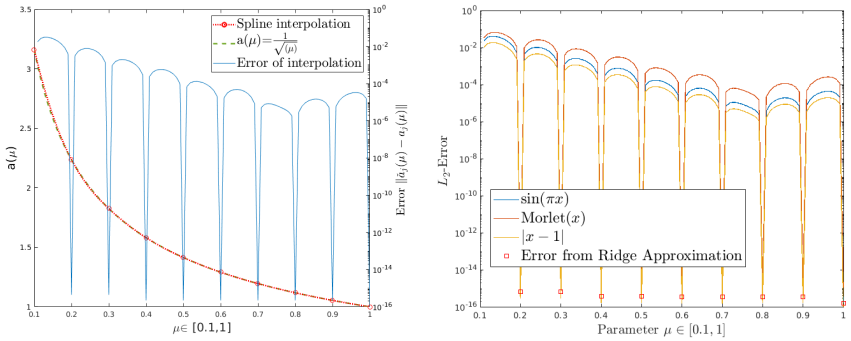


Figure 8. Training directions for the parametric linear transport equation: Spline interpolation of directions (left) and errors for the obtained ridge approximation (right).

5.6. **Conclusions.** As we did much more experiments than we can report here (due to page limitation) let us collect some observations that we have seen and our corresponding conclusions.

- Even though we have seen that the optimization problem arising from the Linear/Ridge approximation is a challenging task, the particle grid algorithm often works very well.
- For approximating a given function, the performance seems to be better for smooth functions. However, the algorithm yields also good results for non-continuous or multivariate functions, even though machine accuracy is harder to reach then.
- By treating time "just as another variable", the presented approach can handle stationary and instationary problems in the same manner.
- Choosing the number of particles sufficiently large, we were always able to reach machine accuracy.
- The algorithm is able to detect if a linear approximation is already sufficient to reach a desired accuracy. Hence, the scheme is robust in the considered cases of fast and of slow decay of the Kolmogorov N-width. We anticipate that this is restricted to (P)PDEs with linear characteristics.

6. Outlook

The above described results of our numerical experiments seem to indicate that this path might be continued. Of course, we are aware that research in several directions is required, e.g.

- the introduced particle grid algorithm is based upon the particle swarm heuristics. There is no rigoros convergence analysis, which is a significant drawback in particular compared to linear RBMs, where online efficiency and a posteriori error control is certified. One might think of using (stochastic) gradient-descent methods in combination with backpropagation/automatic differentiation as known from the training of neural networks. Another option might be to start by the particle swarm algorithm and then use the result as starting point for a decent method.
- in the current form, the particle grid algorithm is not yet online efficient, which would be required for using it within a multi-query and/or realtime environment. Also here, techniques from neural networks might help.
- last, but not least, we did not focus on the training of Φ_N and $\mathcal{V}_M$, but merely viewed them as being given.

References

[1] M. Bachmayr and A. Cohen. Kolmogorov widths and low-rank approximations of parametric elliptic PDEs. *Math. Comp. 86*, 32(304):701–724, 2017.

[2] P. Binev, A. Cohen, W. Dahmen, R. DeVore, G. Petrova, and P. Wojtaszczyk. Convergence rates for greedy algorithms in reduced basis methods. *SIAM J. Math. Anal.*, 43(3):1457–1472, 2011.

[3] F. Black, P. Schulze, and B. Unger. Projection-based model reduction with dynamically transformed modes. *ESAIM: Math. Model. Numer. Anal.*, 54(6):2011–2043, Oct 2020.

[4] A. Bonito, A. Cohen, R. DeVore, D. Guignard, P. Jantsch, and G. Petrova. Nonlinear methods for model reduction. *ESAIM: M2AN*, 55(2):507–531, 2021.

[5] A. Buffa, Y. Maday, A. T. Patera, C. Prud'homme, and G. Turinici. *A priori* convergence of the greedy algorithm for the parametrized reduced basis method. *ESAIM Math. Model. Numer. Anal.*, 46(3):595–603, 2012.

[6] M. D. Buhmann and A. Pinkus. Identifying linear combinations of ridge functions. *Adv. Appl. Math.*, 22(1):103–118, 1999.

[7] J. Cea. Approximation variationnelle des problèmes aux limites. *Annales de l'Institut Fourier*, 14(2):345–444, 1964.

[8] P. Clément. Approximation by finite element functions using local regularization. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, 9(R2):77–84, 1975.

[9] P. G. Constantine, A. Eftekhari, J. Hokanson, and R. Ward. A near-stationary subspace for ridge approximation. *Comput. Methods Appl. Mech. Engrg.*, 326(1):402–421, Nov 2017.

[10] N. Dal Santo, S. Deparis, and L. Pegolotti. Data driven approximation of parametrized pdes by reduced basis and neural networks. *J. Comput. Phys.*, 416:109550, 2020.

[11] V. Ehrlacher, D. Lombardi, O. Mula, and F.-X. Vialard. Nonlinear model reduction on metric spaces. Application to one-dimensional conservative PDEs in Wasserstein spaces. *ESAIM. Math. Model. Numer. Anal.*, abs/1909.06626(54), 2021.

[12] S. Fresca, L. Dede', and A. Manzoni. A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized pdes. *J. Sci. Comp.*, 87(2):61, 2021.

[13] C. Greif and K. Urban. Decay of the kolmogorov n-width for wave problems. *Appl. Math. Letters*, 96:216 – 222, 2019.

[14] B. Haasdonk. Reduced Basis Methods for Parametrized PDEs — A Tutorial. In P. Benner, A. Cohen, M. Ohlberger, and K. Willcox, editors, *Model Reduction and Approximation*, chapter 2, pages 65–136. SIAM, Philadelphia, 2017.

[15] J. S. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer, 2016.

[16] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[17] A. Kolleck and J. Vybíral. On some aspects of approximation of ridge functions. *J. Approx. Th.*, 194:35–61, 2015.

[18] A. Kolmogorov. Über Die Beste Annäherung Von Funktionen Einer Gegebenen Funktionenklasse. *Annals of Mathematics*, 37(1):107–110, 1936.
[19] G. Kutyniok, P. Petersen, M. Raslan, and R. Schneider. A Theoretical Analysis of Deep Neural Networks and Parametric PDEs. *Constr. Approx.*, 2021.
[20] N. J. Nair and M. Balajewicz. Transported snapshot model order reduction approach for parametric, steady-state fluid flows containing parameter-dependent shocks. *Int. J. Numer. Meth. Eng.*, 117(12):1234–1262, 2019.
[21] M. Ohlberger and S. Rave. Nonlinear reduced basis approximation of parameterized evolution equations via the method of freezing. *C.R. Akad. Sci. Math.*, 351(23):901–906, 2013.
[22] M. Ohlberger and S. Rave. Reduced basis methods: Success, limitations and future challenges. *Proceedings of the Conference Algoritmy*, pages 1–12, 2016.
[23] A. E. Olsson. Particle swarm optimization: theory, techniques and applications, 2011.
[24] A. Pinkus. *n-widths in approximation theory*. Springer-Verlag, Berlin, 1985.
[25] A. Pinkus. *Ridge Functions*. Cambridge University Press, 2015.
[26] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced basis methods for partial differential equations: An introduction*. Springer, Cham; Heidelberg, 2016.
[27] G. Welper. Transformed snapshot interpolation with high resolution transforms. *SIAM J. Sci. Comp.*, 42(4):A2037–A2061, 2020.
[28] J. Xu and L. Zikatanov. Some observations on Babuška and Brezzi theories. *Numer. Math.*, 94(1):195–202, 2003.

Ulm University, Institute for Numerical Mathematics, Helmholtzstr. 18, 89081 Ulm (Germany), {constantin.greif,karsten.urban}@uni-ulm.de

Justus-Liebig-University Giessen, Lehrstuhl Numerische Mathematik, Arndtstr. 2, 35392 Giessen (Germany), Philipp.Junk@math.uni-giessen.de