A comparison study of supervised learning techniques for the approximation of high dimensional functions and feedback control

Mathias Oster[1], Luca Saluzzi[2*], Tizian Wenzel[3]

[1]IGPM, RWTH Aachen, Templergraben 55, Aachen, 52062, Germany.
[2*]Department of Mathematics, Scuola Normale Superiore, P.za dei Cavalieri, 7, Pisa, 56126, Italy.
[3]Department of Mathematics, Universität Hamburg, Bundesstraße 55, Hamburg, 20146, Germany.

*Corresponding author(s). E-mail(s): luca.saluzzi@sns.it;
Contributing authors: oster@igpm.rwth-aachen.de;
tizian.wenzel@uni-hamburg.de;

Abstract

Approximation of high dimensional functions is in the focus of machine learning and data-based scientific computing. In many applications, empirical risk minimisation techniques over nonlinear model classes are employed. Neural networks, kernel methods and tensor decomposition techniques are among the most popular model classes. We provide a numerical study comparing the performance of these methods on various high-dimensional functions with focus on optimal control problems, where the collection of the dataset is based on the application of the State-Dependent Riccati Equation.

Keywords: Optimal Control, High-Dimensionality, Neural Networks, Kernel Methods, Tensor Trains

1 Introduction

Finding surrogates for functions in high dimensions has become one of the key tasks in scientific computing. One interesting example is the approximation of the value function of the optimal control of PDEs, which leads, after semi-discretisation, to

very high-dimensional control problems. In optimal control the value function plays a crucial role as it can provide optimal feedback laws. There are two major ways to calculate the value function. First, one could solve the Bellman or Hamilton-Jacobi-Bellman (HJB) equation [1–3]. For both equations there is a wide range of numerical tools in optimal feedback control, see e.g. [4–19]. Second, simultaneously to the work of Bellman, Pontryagin developed a set of necessary conditions for optimal problems [20], the so-called Pontryagin maximum principle (PMP). This enables a pointwise evaluation of the value function and can thus be used for the recovery of the value function [21–24].

Most classical function approximation tools suffer from the curse of dimensionality, i.e. the exponential growth of complexity with respect to the input dimension. To break the curse various methods have been developed, most prominently neural networks, kernel methods and tensor decomposition techniques. In most applications, these methods are based on empirical Least-Squares methods on a nonlinear model class, known as empirical risk minimization techniques [25, 26] in statistic learning. For example, to approximate a function $f : \Omega \mapsto \mathbb{R}$ defined on some domain $\Omega \subset \mathbb{R}^d$, one minimizes the functional

$$\frac{1}{N} \sum_{i=1}^{N} |f(x_i) - f_M(x_i)|^2$$

for N samples $x_i \in \Omega$ distributed according to some density ρ within some model class $f_M \in \mathcal{M} \subset C(\Omega)$. This means that one needs to be able to access the target function f on given or chosen samples. Due to the high dimensionality of the input space Ω, one usually resorts to nonlinear model classes providing improved expressibility compared to linear models of the same complexity, at the cost of increasingly difficult optimization tasks when minimizing the empirical risk.

One of these nonlinear model classes is using structured representations of polynomials like hierarchical tensor formats, which allow to reduce the number of parameters within the coefficient tensor of a linear ansatz [27]. More precisely, we consider a submanifold in $\otimes_{j=1}^{d} \mathbb{R}^{n_i}$ defined by multi-linear parametrizations. Here we use tensor trains which are a special case of a hierarchical or tree based tensor format [27]. Tensor trains have been invented by [28, 29] and applied to various high-dimensional PDE's [30], however the parametrization has already been already used in quantum physics much earlier. For good surveys we refer to [31–34]. The tensor train representation has appealing properties, making them attractive for treatment of the present problems. For example they contain sparse polynomials, but are much more flexible at a price of a slightly larger overhead, see e.g. [35] for a comparison concerning parametric PDEs.

There has been extensive use of tensor trains in high dimensional optimal control problems [36–38] and stochastic control problems [39–42]. In particular, in this paper we will focus on two specific tensor train approaches: the TT Gradient Cross [43] and the block-sparse tensor train [44]. Cross approximation methods [45–48] have been introduced in order to adjust the sampling sets to reduce the conditioning of the interpolation problem and enhance the accuracy of approximation. TT Gradient Cross makes use of the Cross interpolation to construct efficiently the interpolation indices

and takes into account the information of the gradient of the target function to improve the stability. The block-sparse tensor train approach is exploiting sparsity patterns in the cores of tensor trains to avoid overparametrization by identifying homogeneous degree basis functions.

Another approach is the use of kernel methods [49], which comprise various techniques in numerical approximation, machine learning and scientific computing. These flexible tools allow to work with arbitrarily scattered data in high dimensional space and allow for a convenient mathematical analysis based on reproducing kernel Hilbert spaces.

Probably the most popular and widespread techniques in supervised learning are neural networks. The last decade has seen an incredible development in machine learning since efficient back-propagation and optimization algorithms as well as powerful hardware allowed for very complicated neural network architectures, see [50–53] for an introduction to neural networks from a mathematical perspective. They are employed in virtually all machine learning and scientific computing tasks [54, 55]. Also in optimal control they were used to obtain (sub)optimal feedback laws and surrogates for the value functions [56–68].

Lastly, we like to mention that also sparse polynomial techniques have been successfully employed in the context of high dimensional optimal control [69–71]. However, they are not part of the numerical comparison in this study.

In this paper we conduct a numerical study comparing Tensor Trains, neural networks and kernel methods by approximating high dimensional functions and applying them to value functions from high dimensional optimal control problems. After introducing the general optimal control problem in Section 2, we recall our nonlinear model classes in Section 3. In Section 4 we provide a variety of numerical examples where we compare the performance of the different methods. Section 5 concludes the paper.

2 The Optimal Control Problem

In this section we present the infinite horizon problem and the corresponding synthesis of the feedback law. We first introduce the optimal control framework using the Hamilton-Jacobi-Bellman (HJB) formalism, then we pass to the synthesis of suboptimal feedback laws which will provide the data for the supervised learning techniques. We consider a dynamical system in control affine-form given by

$$\begin{cases} \dot{y}(s) = f(y(s)) + B(y(s))u(s), \quad s \in (0, +\infty), \\ y(0) = x \in \mathbb{R}^d. \end{cases} \tag{1}$$

We denote by $y : [0, +\infty) \to \mathbb{R}^d$ the state of the system, by $u : [0, +\infty) \to \mathbb{R}^m$ the control signal and by $\mathcal{U} = L^\infty([0, +\infty); U)$ the set of admissible controls where $U \subset \mathbb{R}^m$. The system dynamics $f : \mathbb{R}^d \to \mathbb{R}^d$ and $B : \mathbb{R}^d \to \mathbb{R}^d$ are assumed to be $\mathcal{C}^1(\mathbb{R}^d)$ functions.

We introduce the infinite horizon cost functional:

$$J(u(\cdot, x)) := \int_0^{+\infty} r(y(s)) + u^\top(s) R u(s)\, ds\,, \tag{2}$$

where $r : \mathbb{R}^d \to \mathbb{R}^+$ and $R \in \mathbb{R}^{m\times m}$ is a symmetric positive definite matrix. Our aim is to compute an optimal control in feedback form, *e.g.* a control signal fully determined upon the current state of the system. We start by the definition of the value function for a given initial condition $x \in \mathbb{R}^d$:

$$V(x) := \inf_{u\in\mathcal{U}} J(u(\cdot, x))\,, \tag{3}$$

which satisfies the following Hamilton-Jacobi-Bellman PDE for every $x \in \mathbb{R}^d$

$$\min_{u\in U} \left\{ (f(x) + B(x)u)^\top \nabla V(x) + r(x) + u^\top R u \right\} = 0. \tag{4}$$

The HJB PDE (4) is a first-order fully nonlinear PDE defined over $\mathbb{R}^d$, where d represents the dimension of the considered dynamical system. The dimension of the problem may be large and this limitation is known as curse of dimensionality. In the case, *i.e.* $U = \mathbb{R}^m$, the minimization in the equation (4) can be computed explicitly as

$$u^*(x) = -\frac{1}{2} R^{-1} B(x)^\top \nabla V(x)\,, \tag{5}$$

leading to the following version of the HJB PDE

$$\nabla V(x)^\top f(x) - \frac{1}{4} \nabla V(x)^\top B(x) R^{-1} B(x)^\top \nabla V(x) + r(x) = 0\,. \tag{6}$$

In this work, instead of considering directly the high-dimensional HJB PDE (6), we are interested in retrieving an approximation of the value function using supervised learning techniques. Indeed, we will consider an approximation of $V(x)$ in a regression framework, where we assume measurements of the function at sampling points.

We are going to consider a fast, but suboptimal alternative: the State-Dependent Riccati Equation (SDRE). This will allow to generate a synthetic dataset that allows to approximate the value function, leading to the synthesis of a suboptimal control which is still able to stabilize the dynamical system.

2.1 State-Dependent Riccati Equation

The State Dependent Riccati Equation (SDRE) is a powerful mathematical tool that finds widespread applications [72, 73]. Originating from the classical Riccati Equation, the SDRE extends its utility by incorporating state-dependent coefficients, thereby accommodating systems with nonlinearity and time-varying dynamics. The approach relies on sequential resolution of linear-quadratic control problems that stem from progressive linearization of the dynamics along a trajectory.

Let us suppose that the cost functional (2) can be rewritten in the following form

$$J_\infty(u;x) = \int_0^{+\infty} y(t)^\top Q(y) y(t) + u(t)^\top R u(t)\, dt\,, \tag{7}$$

with $Q : \mathbb{R}^d \to \mathbb{R}^{d\times d}, Q(y) \succeq 0\ \forall y \in \mathbb{R}^d$ and the dynamical system expressed in semilinear form

$$\dot{y}(t) = A(y(t))y(t) + B(y(t))u(t) \tag{8}$$
$$y(0) = x\,. \tag{9}$$

Assume the dynamics is linear in the state, *i.e.* $A(y(t)) = A \in \mathbb{R}^{d\times d}$ and $B(y(t)) = B \in \mathbb{R}^{d\times m}$, and that the matrix $Q(y) = Q$ is constant in the state. This is called Linear Quadratic Regulator (LQR) problem. If the pair (A, B) is stabilizable and the pair $(A, Q^{1/2})$ is detectable, then the optimal feedback control is computed via the following formula

$$u(y) = -R^{-1}B^\top P y. \tag{10}$$

Here, $P \in \mathbb{R}^{d\times d}$ is the unique positive definite solution of the Algebraic Riccati Equation (ARE)

$$A^\top P + PA - PBR^{-1}B^\top P + Q = 0\,.$$

Essentially, the SDRE technique extends this approach introducing the dependence on the current state, that is,

$$u(y) = -R^{-1}B^\top(y)P(y)y\,, \tag{11}$$

where $P(y)$ now is the solution of a state-dependent ARE

$$A^\top(y)P(y) + P(y)A(y) - P(y)B(y)R^{-1}B^\top(y)P(y) + Q = -Q(y), \tag{12}$$

where $A(y)$ and $B(y)$ are fixed at the state y. This procedure can be iterated along the trajectory, solving sequentially (12) as the state $y(t)$ is evolving in time.

Assuming suitable stability hypothesis, it is possible to show that the closed loop dynamics generated by the feedback law (11) are locally asymptotically stable (we refer to [74] for the exact statement and further details).

3 Machine learning methods

3.1 Machine learning and least-squares loss

In this article we focus on the use of supervised machine learning techniques for high-dimensional functions and optimal control problems. In each of the following numerical example we aim to approximate a target function f. One way to do so is to minimize

an empirical L^2 loss between the target function and a nonlinear model class, i.e. we are considering a problem of the form

$$\min_{f_M \in \mathcal{M}} \frac{1}{N} \sum_{j=1}^{N} |f(x_j) - f_M(x_j)|^2, \tag{13}$$

where x_j are samples in Ω and $\mathcal{M}$ is the model class. This allows us to use random samples. Another route is taken by the TT-Cross algorithm introduced in the following. There an active learning strategy is employed to reduce the sample complexity.

All methods which will be introduced in the following do employ a global function approximation, which is in contrast to localized methods such as as finite elements. Furthermore, all model classes will be nonlinear except the kernel method.

3.2 Tree Based Tensor Representation - Tensor Trains

For the approximation of the value function (3), we define a nonlinear model class to circumvent the curse of dimensionality. To this end, we choose an underlying finite dimensional subspace for the approximation of the sought value function. For the present purpose we take a space $\Pi_{i,n_i} = \operatorname{span}\{\psi_{i_1}, \dots, \psi_{i_d}\}$ of one-dimensional polynomials of degree smaller than n_i and consider the tensor product of such polynomial spaces

$$\mathcal{V}_p := \Pi_{1,n_1} \otimes \cdots \otimes \Pi_{d,n_d}.$$

This is a space of multivariate (tensor product) polynomials with bounded multi-degree. Its elements $v \in \mathcal{V}_p$ can be represented as

$$v(x_1, \dots, x_d) = \sum_{i_1,\dots,i_d=0}^{n_1,\dots,n_d} c_{i_1,\dots,i_d} \psi_{i_1}(x_1) \cdots \psi_{i_d}(x_d), \tag{14}$$

exhibiting that $c \in \mathbb{R}^{n_1,\dots,n_d}$ suffers from the curse of dimensionality. For the sake of readability we will henceforth write $c[i_1, \dots, i_d] = c_{i_1,\dots,i_d}$ and say that c is an order d tensor.

The tensor train decomposition aims to represent an order d tensor by a sequence of order 3 tensors, connected by contractions. This means that we represent c by $U_1 \in \mathbb{R}^{n_1,r_1}$, $U_2 \in \mathbb{R}^{r_1,n_2,r_2}, \dots, U_{d-1} \in \mathbb{R}^{r_{d-2},n_{d-1},r_{d-1}}$ and $U_d \in \mathbb{R}^{r_{d-1},n_d}$ such that

$$c[i_1, \dots, i_d] = \sum_{j_1=1}^{r_1} \cdots \sum_{j_{d-1}=1}^{r_{d-1}} U_1[i_1, j_1] U_2[j_1, i_2, j_2] \dots U_d[j_{d-1}, i_d]. \tag{15}$$

The TT-rank is introduced as the element wise smallest tuple $\mathbf{r} = (r_1, \dots, r_{d-1})$ such that a decomposition of the form (15) exists. The TT-rank is well defined and, denoting $r = \max\{r_i\}$ and $n = \max\{n_i\}$, the tensors of fixed TT-rank form a smooth manifold of dimension $\mathcal{O}(dnr^2)$, which means that for fixed ranks the dimension of the manifold does increase linearly with the order d. Quadratic functions [36] or weakly correlated

Gaussian functions [75], for example, admit an approximation with TT-rank growing at most polynomial in d and poly-logarithmic in the approximation error.

Observing that the component tensors U_i are connected via a single contraction/summation to U_{i-1} and U_{i+1}, we can represent the decomposition in a graph, by setting the components U_i as nodes and indicate contractions by links between the nodes. In the next step we plug the TT-decomposition of the coefficient ten-

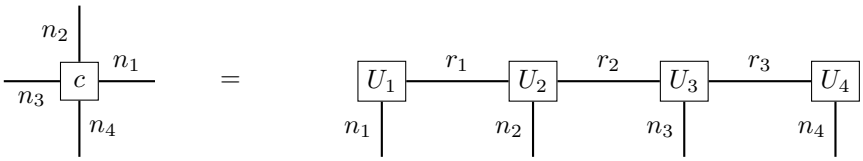

Fig. 1 Graphical representation of a TT representation of c in four variables.

sor (15) into the representation in (14). To this end we introduce the short form $\Psi_i(x_i) = [\psi_{i_1}(x_i), \dots, \psi_{i_d}(x_i)] \in \mathbb{R}^{n_i}$. Then

$$v(x_1, \dots, x_d) = \sum_{i_1,\dots,i_d}^{n_1,\dots,n_d} \sum_{j_1,\dots,j_{n-1}}^{r_1,\dots,r_{n-1}} U_1[i_1, j_1] U_2[j_1, i_2, j_2] \dots U_d[j_{d-1}, i_d] \\ \big(\Psi_1(x_1)\big)[i_1] \big(\Psi_2(x_2)\big)[i_2] \cdots \big(\Psi_d(x_d)\big)[i_d], \quad (16)$$

which means that every open index of the TT representation is contracted with the one-dimensional basis functions. This representation is known as Functional Tensor Train (FTT) format of the function v. The graphical representation of this tensor network is given in Figure 2. Note that any basis can be chosen for Ψ_i. In this paper

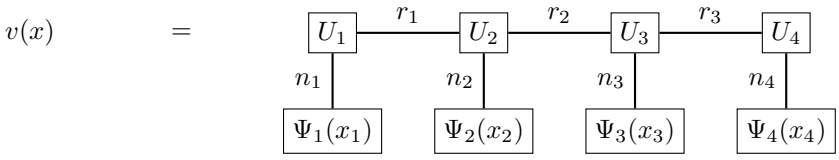

Fig. 2 Graphical representation of TT tensor train induced polynomial in four variables.

we use a set of orthonormal polynomials. In this case, we have a Parseval formula providing a norm equivalence between the function space and the Frobenius norm of the coefficients, which guarantees stability of our representations.

It turns out, that optimization procedures in this TT format can be solved by consecutively optimizing one component U_l while the others are fixed. This alternating Least-Squares (ALS) algorithm converges at least to a local minimum [76].

3.2.1 Block-Sparse Tensor Trains

One important observation is that in general, tensor trains will parameterize polynomials with high-mixed degree which might lead to numerical instabilities if the

sought function has a bounded maximal degree. To overcome this superfluous degrees of freedom one can employ the so-called block sparse tensor trains [44].

As it turns out, homogeneous polynomials of degree $\tilde{g}$ exhibit a representation as tensor trains for which the core admit a sparse representation with block sizes $\rho_{k,\tilde{g}}$ which also provide rank bounds. Very importantly these block sparse structures are preserved under essential tensor train manipulation as TT-SVD and rounding. Furthermore, the ALS algorithm can be restricted to respect these sparsity patterns. By introducing an extra index in the last core one can also parameterize non-homogeneous polynomials in a block sparse fashion.

Let us give a small example on how the cores will look like for the block-sparse TTs.

Example 1 (*Block Sparsity*)**.** *Let $p = 4$ and $g = 3$ be given and let c be a tensor train such that $Lc = gc$. Then for $k = 2, \dots, d-1$ the component tensors C_k of c exhibit the following block sparsity (up to permutation). For indices i of order r_{k-1} and j of order r_k*

$$C_k(i,1,j) = \begin{pmatrix} * & 0 & 0 & 0 \\ 0 & * & 0 & 0 \\ 0 & 0 & * & 0 \\ 0 & 0 & 0 & * \end{pmatrix} \quad C_k(i,2,j) = \begin{pmatrix} 0 & * & 0 & 0 \\ 0 & 0 & * & 0 \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 \end{pmatrix}$$
$$C_k(i,3,j) = \begin{pmatrix} 0 & 0 & * & 0 \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad C_k(i,4,j) = \begin{pmatrix} 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Another structural assumption that frequently can be found in application is a certain grouping of variables, i.e. the function can be written as a sum of sub-functions each of which depend only on few variables. To make use of this, the following notion of locality is introduced:

Definition 1 ([44])**.** *Let $u \in W_g^d$ be a homogeneous polynomial and B be the symmetric coefficient tensor. We say that u has a variable locality of K_{loc} if $B(\ell_1, \dots, \ell_g) = 0$ for all $(\ell_1, \dots, \ell_g) \in \mathbb{N}_d^g$ with*

$$\max\{|\ell_{m_1} - \ell_{m_2}| \,:\, m_1, m_2 = 1, \dots, g\} > K_{\text{loc}}.$$

Note that the locality and degree bound will give rank bounds. All in all, we will employ this block-sparse tensor train to reduce the complexity of the ansatz space. We will comment in the numerical examples, where this ansatz will be too restrictive and where it can provide benefits to the usual TTs.

3.2.2 TT-Gradient Cross

Given a target function V and its FTT representation (16) $\tilde{V}$, the TT-Gradient Cross algorithm aims to solve for a given sample points $\{x_i\}_{i=1}^N$ and a dataset $\{V(x_i), \nabla V(x_i)\}_{i=1}^N$ the following regression problem

$$\min_{U_1,\dots,U_d} \sum_{i=1}^{N} |\tilde{V}(x_i) - V(x_i)|^2 + \lambda \|\nabla \tilde{V}(x_i) - \nabla V(x_i)\|^2, \tag{17}$$

where λ is a parameter tuning the gradient information. The problem can be attacked using fast algorithms based on the so-called *cross interpolation* [45]. Given a prefixed set of collocation points $X_1 \times \ldots \times X_d$, we apply an alternating direction strategy solving sequentially least square problems. At the $k-$th iteration our goal is to find interpolation sets $\overline{X}_{<k} \subset X_1 \times \cdots \times X_{k-1}$ and $\overline{X}_{>k} \subset X_{k+1} \times \cdots \times X_d$ with r_{k-1} and r_k points, respectively. Let us suppose that in the k-th step the sets $\overline{X}_{<k}$ and $\overline{X}_{>k}$ are given. We point out that the number of unknowns in U_k and the cardinality of the set $\overline{X}_{<k} \oplus X_k \oplus \overline{X}_{>k}$ is equal to $r_{k-1} n_k r_k$. Solving the least square problem related to actual sampling points, one can compute the current U_k and thanks to pivoting techniques (for example the *maxvol* method [77]), it is possible to select the next sampling sets $X_{<k+1}$ and $X_{>k-1}$ as subsets of $X_{<k} \oplus X_k$ and $X_k \oplus X_{>k}$. This step can be iterated for all $k = 1, \dots, d$ and the TT cores are updated accordingly until the algorithm converges. In this case we claim that the method converges if the norm of the difference of the coefficients in Frobenius norm computed in two consecutive steps is below a certain threshold denoted as tol_{stop}.

In the context of optimal control problems, the target function V is the value function and the surrogate model $\tilde{V}$ helps for a fast synthesis of feedback controls. Indeed, the optimal control is given by

$$u(x) = -\frac{1}{2} R^{-1} B(x)^\top \nabla V(x),$$

where R and $B(x)$ have been introduced in Section 2. Computed an approximation of the FTT representation (16) by the TT Gradient Cross, in the online phase we need to compute the gradient of the surrogate model obtaining the feedback control

$$\tilde{u}(x) = -\frac{1}{2} R^{-1} B(x)^\top \nabla \tilde{V}(x).$$

The TT format enables to compute the gradient in $O(dnr^2)$ operations per point, resulting in an expedited synthesis of the feedback control. For a detailed description of the method and its application we refer to [43].

3.3 Kernel methods

Another popular method in machine learning are summarized as kernel methods. This class of methods revolves around the use of a kernel k, which is a symmetric function $k : \Omega \times \Omega \to \mathbb{R}$, that satisfies some definiteness properties like strict positive definiteness, i.e. the kernel matrix $(k(x_i, x_j))_{i,j=1}^n$ is positive definite for any choice of pairwise distinct points $\{x_i\}_{i=1}^n$ and any $n \in \mathbb{N}$. Some popular examples are the

Gaussian kernel or the exponential kernel,

$$\begin{aligned} k_{\text{Gaussian}}(x,y) &= \exp(-\|x-y\|^2), \\ k_{\exp}(x,y) &= \exp(-\|x-y\|) \end{aligned} \tag{18}$$

which are radial basis function kernels in $\mathbb{R}^d$ for any $d \in \mathbb{N}$. It turns out, that every strictly positive definite kernels gives rise to a unique reproducing kernel Hilbert spaces (RKHS) $\mathcal{H}_k(\Omega)$, which allows for a thorough theoretical analysis. Under mild assumptions on the domain Ω, e.g. a Lipschitz boundary, these RKHS can frequently characterized in terms of Sobolev spaces. For example, the RKHS of the exponential kernel from Eq. (18) is norm equivalent to the Sobolev space $H^{(d+1)/2}(\Omega)$, while the RKHS of the Gaussian kernel consists of analytic functions and a full characterization is more sophisticated. A representer theorem for kernel approximation [26] states that the optimal solution for the MSE loss task of Eq. (13) can be found as

$$s_X(\cdot) = \sum_{j=1}^{M} \alpha_j k(\cdot, x_j), \tag{19}$$

whereby the coefficients $\{\alpha_j\}_{j=1}^M$ can be frequently computed directly. Such kernel models are used for statistical learning [26], numerical approximation [49], PDE approximation [78] and machine learning [79], among others. Recent machine learning research also aims at modifying the kernel, in order to obtain data-adapted or deep kernel models [80–82]. In numerical approximation, based on assumptions like $f \in \mathcal{H}_k(\Omega)$ or also weaker ones, sharp error estimates for the residual $f - s_X$ on some Lipschitz domain $\Omega \subset \mathbb{R}^d$ can be derived in various norms, e.g. $\|\cdot\|_{L^2(\Omega)}, \|\cdot\|_{L^\infty(\Omega)}$ or even Sobolev norms [83, 84]. Similar to other approximation methods, such error estimates suffer the curse of dimensionality, however target data adapted point choices $X \subset \Omega$ offer the possibility to break it [85]. Though the computation of the optimal weights $\{\alpha_j\}_{j=1}^M$ can be done explicitly, this can be challenging from the computational point of view for large amount of data points $M \gg 1$. Thus recent research aims at scaling kernel methods to large amounts of data, e.g. via iterative preconditioned training methods and efficient implementations [79, 86].

3.4 Neural Networks

The most prominent machine learning tools are (deep) neural networks [87], that are used especially for high-dimensional approximation tasks like image recognition or image generation. Although NN techniques proved to be very powerful for real world applications, it turns out that they pose difficulties to obtain quantitative results for approximation theoretical bounds due to the iterative and non-convex training procedures. There is a plethora of sophisticated architectures and structures available like convolutional neural networks, residual neural network and recurrent neural networks aiming on different aspects of machine learning tasks as image and speech recognition. For high-dimensional unstructured data for function regression, we focus in the forthcoming on plain feedforward (residual) neural networks, because neither a spatial

structure (as in image data) nor a temporal structure (as in time series) is assumed. Standard feedforward networks are given as a concatenation of L consecutive layers, each described by a simple nonlinear transform:

$$f_l(x) = \sigma(W_l x + b_l). \tag{20}$$

As it turns out, the feedforward NN can suffer some numerical instabilities that can be partly elevated by the so-called residual network architecture where each layer takes the form

$$f_l(x) = x + \sigma(W_l x + b_l). \tag{21}$$

The function σ is called activation function, which is a pointwise acting nonlinear function that introduces nonlinearity into the approach. A prominent example is the ReLU function $\sigma(x) = \max(x, 0)$ or variants like Leaky ReLU, SeLU, GeLU among others. The matrices W_l and the bias vectors b_l constitute the parameters of the neural network, which can be optimized. Due to the highly-nonlinear structure of the NN, a closed form solution for the optimal weights $\{W_l\}_{l=1}^L$ and biases $\{b_l\}_{l=1}^L$ is usually infeasible. Thus, in practice neural networks are optimized using gradient descent like mini-batch optimization strategies, where the Adam optimizer [88] which uses adaptive moment estimations is probably the most popular optimizer. In order to improve generalization for neural networks, regularization strategies like DropOut and Early Stopping are available, although modern neural network models are frequently overparameterized, mitigating the need for explicit regularization. Despite there are error bounds available which elucidate especially the benefits of deep neural networks [89], they are rather of constructive nature. As it is highly unlikely that the stochastic gradient based optimization realizes such constructions, these error bounds are usually not practical. It is remarkable, that recently connections between the training of neural network and kernel methods where found: In the limit of large width neural networks, the training behaviour of neural networks is linearized in parameter space and can thus be described with help of the neural tangent kernel [90].

4 Numerical tests

In this section, we embark on an exploration and comparison of the supervised learning techniques introduced in the previous sections through a series of numerical tests. In all the examples we employ sample-based algorithms for function regression. Given a set of sample points $\{x_j\}_{j=1}^n$, we measure the error between the prediction $s(x_j)$ and the true target value $y_j = f(x_j)$ considering the following relative error in norm 2:

$$err_2 = \sqrt{\frac{\sum_{j=1}^n |y_j - s(x_j)|^2}{\sum_{j=1}^n |y_j|^2}}. \tag{22}$$

In the tables below, we add *train* or *test* in the subscript in order to define whether the error was measured with respect to the training or the test set. In the last example

we will consider also the error in the computation of the total cost obtained along the optimal trajectories. To this end, we introduce $cost_{SDRE}(x)$ as the total cost computed using the SDRE feedback and $cost_{surr}(x)$ as the total cost calculated using the gradient of the surrogate to the value functional as control signal in (5). Finally, we define the error in the total cost at a given initial condition $x \in \mathbb{R}^d$:

$$err_{cost}(x) = |cost_{SDRE}(x) - cost_{surr}(x)|.$$

Remark 1. *Notice that the methods were running on different machines and with different languages. Especially, the TT cross methods is based on a very performative implementation, while the block-sparse TT and kernel approximation did not use such tuned implementations. Moreover, the TT Gradient Cross and its applications are Matlab-based, while the rest of the codes are written in Python and Julia.*

4.1 Low against high rank

First, we consider the following three test cases

a) $f(x) = \exp(-\sum_{i=1}^{d} x_i/(2d)), \quad x \in [-1,1]^d,$
b) $f(x) = \exp(-\prod_{i=1}^{d} x_i), \quad x \in [-1,1]^d,$
c) $f(x) = \exp(-\prod_{i=1}^{d} x_i), \quad x \in [0,2]^d.$

We note that $f(x) = \exp(-\sum_{i=1}^{d} x_i/(2d))$ has a rank one decomposition in a functional tensor train in contrast to the other two examples. The functions reported in the cases (b) and (c) are not low-rank and this is reflected in the numerical results. The difference between cases (b) and (c) are the different domains. Test case (b) is set in $[-1,1]^d$ and the function is almost 1 for a significant part of the domain, while in case (c) the function takes values between 0 and 1.

For each of three test cases, the methods described in Section 3 were applied. The approximation take place in dimension $d = 16$ and the tables for each methods display the train and test error as well as the degrees of freedom (DoFs), the CPU time and the sample size. The test error is based on a 10^4 uniform random samples in the respective domain.

For the TT Gradient Cross, the space domain is discretized with 7 Legendre-Gauss nodes for each direction. We fix the stopping error tol_{stop} equal to 10^{-5}. For the block sparse TT we consider degree 7 Legendre polynomials and a locality to match the degrees of freedom of the TT Cross results. Throughout the experiments, the stopping criterion was set to 10^{-11} or a maximal ALS iteration. For the kernel approximation, we considered 5000 low discrepancy points within the given domain, and use the quadratic Matérn kernel with shape parameter in $\{\frac{1}{2\sqrt{d}}, \frac{1}{4\sqrt{d}}, \frac{1}{8\sqrt{d}}\}$ and without explicit regularization. For the neural network, as standard setup and training was used: The neural networks used three fully connected layers of width 512, and the training was done using mini batches of size 128, the Adam optimizer and an initial learning rate of 1e-3.

As the function considered in case (a) has an exact TT decomposition with rank 1, it can be approximated arbitrary well by the TT Gradient Cross, as described by both

$err_{train,2}$ and $err_{test,2}$. Also the low number of samples and the small CPU times are due to the low rank structure of the function. Similar, the block-sparse TT methods behaves very well. The cases (b) and (c) are more tricky and the magnitudes of the error indicators drops of several orders with respect to the first case. Especially for the third case for which the number of training samples increases drastically, obtaining just a testing error of order $O(10^{-1})$ for the block sparse approach and $O(10^{-2})$ for the TT Gradient Cross. For the block sparse TT format, an additional difficulty is due to the functions (b) and (c) not satisfying a locality bound. As no rounding was employed in the block sparse format, the number of degrees of freedom might be an overestimation. The kernel methods also are able to recover the chosen functions, despite the errors are not as small. Especially the function (c) is the most challenging one with an error in $O(10^{-1})$. This numbers are an indicator, that the kernel approximation using a radial kernel together with uniformly distributed samples is clearly affected from the curse of dimensionality. Note that the kernel methods use significantly less samples and degrees of freedom. The neural networks achieve an error of $O(10^{-1})$ or $O(10^{-2})$, which is frequently sufficient for machine learning purposes, however here not on par with the kernel methods or the tensor trains. Due to the stochastic iterative training procedure of the neural networks, some time had to be spent to find suitable hyperparameters, while there are likely still further options to tune the NN and its training. Overall, the TT Cross approximation provides the best approximations.

		$err_{train,2}$	$err_{test,2}$	DoFs	# train sample	CPU train (s)	CPU test (s)
TT Cross	(a)	6.97e-16	7.60e-16	420	1253	0.11	0.47
	(b)	1.56e-7	3.16e-8	3612	13937	0.23	0.33
	(c)	7.66e-4	2.76e-2	35196	293265	3.45	0.41
BSTT	(a)	6.90e-11	1.28e-09	434	1900	27.5	0.15
	(b)	4.40e-5	1.54e-4	3689	14000	1262.5	0.60
	(c)	6.03e-3	9.05e-2	27935	100000	43973.7	28.0
NN	(a)	1.23e-2	1.56e-2	797185	5000	46.48	0.13
	(b)	9.39e-3	1.41e-2	797185	5000	56.04	0.12
	(c)	6.32e-2	1.14e-1	797185	5000	32.12	0.11
Kernel	(a)	4.76-12	9.35e-6	5000	5000	2.30	2.54
	(b)	2.98e-10	9.63e-4	5000	5000	2.39	2.60
	(c)	2.98e-10	3.61e-1	5000	5000	2.36	2.55

Table 1 Results of the different methods for Test 4.1 ($d = 16$).

4.2 Regularity test

For the next test, we are interested in the effects of non-differentiablities of the target functions on the expressability of our model classes. To that end, we consider the following family of functions

$$f(x) = \lambda_0\|x\|^2 + \lambda_1\|x - y_1\| + \lambda_2\sqrt{\|x - y_2\|}, \quad x \in [-1, 1]^d,$$

where $y_1 = (0.5, \ldots, 0.5)$ and $y_2 = (-0.5, \ldots, -0.5)$ and consider the cases of $\lambda = (\lambda_0, \lambda_1, \lambda_2) \in \{(1,0,0), (1,0.5,0), (1,0.5,0.5), (0,0.5,0), (0,0,0.5)\}$. Again, all the method described in Section 3 are applied.

For the TT Gradient Cross, the intervall $[-1,1]$ is discretized using 7 Legendre-Gauss nodes per direction and a stopping tolerance of $tol_{stop} = 10^{-5}$ is fixed. The block sparse TT where chosen to have the same complexity as the TT Cross approximation. Both the kernel approximation and the neural network are using the same setup as in the previous example. For every method, the test error is evaluated on 10^4 random samples in $[-1,1]^d$.

In this example one can see that, as expected, non-differentiablilities are more difficult to approximate with all of our methods. Here, again the TT Cross method outperforms the other model classes providing results with improved accuracy of two orders of magnitude compared to the block sparse format and the kernel methods and up to four orders of magnitude compared to the NN approach. The TT methods can recover the squared norm almost exactly. Interestingly, there is almost no difference between the approximation error for $f(x) = \|x\|$ and $f(x) = \sqrt{\|x\|}$ within the different model classes respectively. This is likely due to the fact that the non-differentiability is very localized and thus not properly detected by the 10^4 randomly sampled test points. This is displayed in Figure 4.2. We note that TT-Cross algorithm is able to mimic the behaviour around the kink, while the kernel and NN approximation simply leave out this localized kink.

	λ_0	λ_1	λ_2	$err_{train,2}$	$err_{test,2}$	DoFs	# train samples
TT Cross	1	0	0	7.50e-16	9.64e-16	924	1994
	1	0.5	0	7.67e-7	1.57e-6	6608	14879
	1	0.5	0.5	1.49e-6	3.33e-6	13293	32235
	0	0.5	0	2.85e-7	2.19e-6	7322	16776
	0	0	0.5	5.83e-7	2.56e-6	7308	17334
BSTT	1	0	0	9.62e-12	4.57e-11	96	1994
	1	0.5	0	1.41e-05	5.02e-5	5069	10000
	1	0.5	0.5	2.57e-05	7.4e-5	12786	32000
	0	0.5	0	8.56e-05	3.4e-4	7378	16000
	0	0	0.5	8.12e-05	3.1e-4	7378	17334
NN	1	0	0	1.86e-2	3.13e-2	797185	5000
	1	0.5	0	1.501-2	2.62e-2	797185	5000
	1	0.5	0.5	9.82e-3	2.17e-2	797185	5000
	0	0.5	0	4.78e-3	9.05e-3	797185	5000
	0	0	0.5	8.41e-3	1.11e-2	797185	5000
Kernel	1	0	0	9.74e-8	1.07e-4	5000	5000
	1	0.5	0	9.01e-9	2.83e-4	5000	5000
	1	0.5	0.5	3.75e-7	2.18e-4	5000	5000
	0	0.5	0	4.34e-13	1.14e-3	5000	5000
	0	0	0.5	4.28e-13	1.08e-3	5000	5000

Table 2 Results for the different methods for Section 4.2 ($d = 16$).

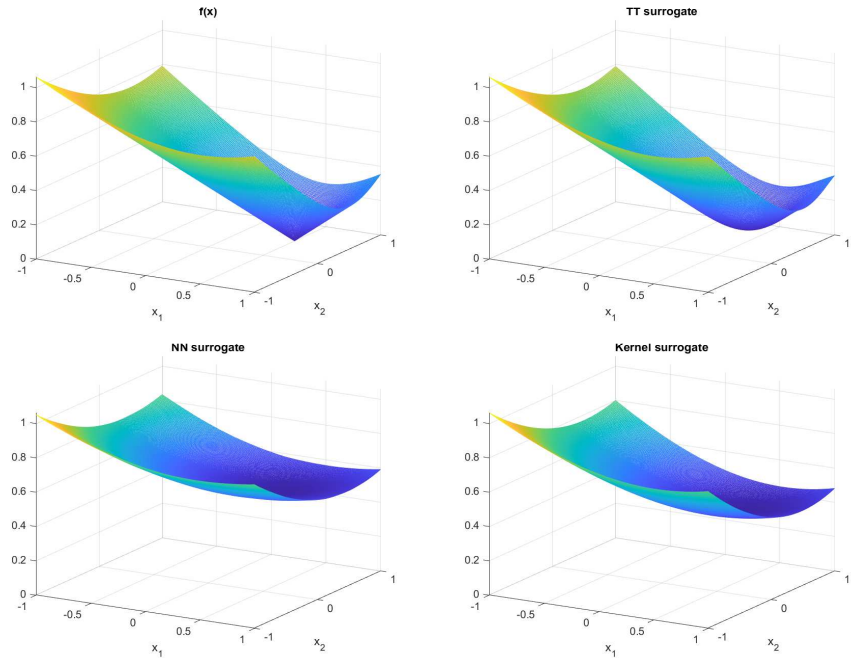

Fig. 3 Function and surrogate model for $d = 16$ and $\lambda = (0, 0.5, 0)$ for the different methods on the plane $(x_1, x_2, 0.5, \ldots, 0.5)$. The localized kink is only fitted properly by the TT approximation.

4.3 Academic Optimal Control Example

In this numerical example we want to deal with the application of the mentioned techniques to an "academic" optimal control problem. Following the idea described in [91], our aim is to consider a control problem with a prefixed value function V. Now, consider the dynamical system

$$\dot{x} = u, \quad x(0) = x_0 \tag{23}$$

and the cost functional

$$J(x_0, u) = \int_0^{+\infty} r(x(t)) + \frac{1}{2}\|u(t)\|^2 \mathrm{d}t, \tag{24}$$

where $r(x)$ will be chosen to obtain the value function V. In this case the HJB equation reads

$$-\frac{1}{2}\|\nabla V(x)\|^2 + r(x) = 0.$$

Then, choosing exactly $r(x) = \frac{1}{2}\|\nabla V(x)\|^2$, we obtain that the optimal control problem (23)-(24) admits V as value function. The aim of this example is twofold: We want to test the accuracy of the SDRE approach in the approximation of the feedback law

and we want to compare the supervised learning techniques varying the dimension of the problem.

SDRE approximation

In this paragraph we focus on the accuracy of the SDRE approximation of the optimal control problem (23)-(24). The dynamical system (23) can be written easily in semilinear form (9) with $A(x) = 0_d$ and $B(x) = I_d$, where $0_d \in \mathbb{R}^{d\times d}$ is a matrix of all zeros and $I_d \in \mathbb{R}^{d\times d}$ is the identity matrix. The cost functional (24) is then written in a state-dependent quadratic form (7) with $R(x) = \frac{1}{2}I_d$ and fixing $Q(x)$ such that

$$x^\top Q(x)x = r(x) = \frac{1}{2}\|V(x)\|^2 \ .$$

In this case the associated SDRE for $x \in \Omega$ reads

$$-2P(x)^2 + Q(x) = 0$$

with solutions $\pm\sqrt{Q(x)/2}$. Since we are interested in the positive definite solution, we choose $P(x) = \sqrt{Q(x)/2}$. We note that the choice of $Q(x)$ is crucial (see [92, 93] for a discussion on the importance of the representation of the system). Indeed, choosing

$$Q(x) = \frac{1}{2}\begin{bmatrix} |\partial_{x_1}V|^2/x_1^2 & & \\ & \ddots & \\ & & |\partial_{x_d}V|^2/x_d^2 \end{bmatrix},$$

the solution of the SDRE reads

$$P(x) = \frac{1}{2}\begin{bmatrix} \partial_{x_1}V/x_1 & & \\ & \ddots & \\ & & \partial_{x_d}V/x_d \end{bmatrix},$$

and the corresponding feedback control can be computed via the formula (10):

$$u(x) = -\frac{1}{2}P(x)x = -\frac{1}{4}\nabla V,$$

retrieving exactly the formula of the optimal feedback control (5). In this specific example the exactness of the SDRE approach comes only in case we are able to extract the norms of the single partial derivatives $|\partial_{x_i}V|^2$ from the the given cost function $r(x) = \frac{1}{2}\|V\|^2$.

Supervised learning approximation

Now let us compare the different techniques on a specific example. Let us choose the value function of the form

$$V(x) = \|x\|^2 (e^{-\frac{\|x-\mu_1\|^2}{\sigma_1^2}} + e^{-\frac{\|x-\mu_2\|^2}{\sigma_2^2}}), \tag{25}$$

where $\mu_1, \mu_2 \in \mathbb{R}^d$ and $\sigma_1, \sigma_2 \in \mathbb{R}^+$ are parameters denoting respectively the means and the standard deviations of the corresponding gaussian functions. We can see immediately that the value function can be expressed in the form $V(x) = x^\top P(x) x$ defining

$$P(x) = \begin{bmatrix} (e^{-\frac{\|x-\mu_1\|^2}{\sigma_1^2}} + e^{-\frac{\|x-\mu_2\|^2}{\sigma_2^2}}) & & \\ & \ddots & \\ & & (e^{-\frac{\|x-\mu_1\|^2}{\sigma_1^2}} + e^{-\frac{\|x-\mu_2\|^2}{\sigma_2^2}}) \end{bmatrix}.$$

First of all, we fix $\mu_1 = 0 \cdot \mathbf{1}$, $\mu_2 = 0.5 \cdot \mathbf{1}$, $\sigma_1 = \sigma_2 = 1$, where $\mathbf{1} \in \mathbb{R}^d$ is a vector with all ones. The dimension of the problem d will vary in the set

$$D = \{3, \dots, 16\}$$

and the initial condition x_0 will be taken randomly in the set $\Omega = [-1, 1]^d$. As the variance is dimension independent one would expect the function to become smoother with increased dimension. Indeed, the TT Cross method has an improving approximation error with respect to the dimensions.

The TT-Gradient Cross is implemented using a stopping tolerance 10^{-5} and 7 Legendre-Gauss polynomials per dimension. We see that as we increase the dimension the error improves, but stagnating around the order 10^{-3}. On the hand, the CPU is keeping an almost constant behaviour of order $O(1)$, due to the fact that the TT rank is at most 5 for all the considered dimensions.

For the kernel approximation, we employ the quadratic Matérn kernel using 10000 low-discrepancy points within $[-1, 1]^d$ and choose the shape parameter as $\frac{1}{2\sqrt{d}}$. The kernel approximation is the best approximation up to dimension 5, while its error increases as the dimension grows. The behaviour of the CPU time is more or less constant of $O(10)$, as it mainly depends on the number of sample points. Finally, the NN approach, which uses the same setup as in the previous examples, obtains the worst results in terms of both indicators, with an increasing error between $O(10^{-2})$ and $O(10^{-1})$ as the dimension d of the problem grows and a CPU time varying between $O(10)$ and $O(10^2)$.

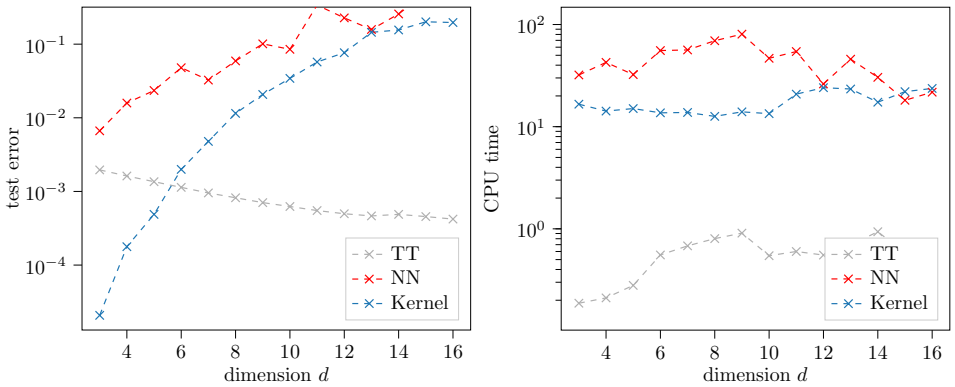

Fig. 4 Visualization of the test error (22) (left) and the CPU time (right) for approximation of the value function (25) depending on the dimension d (x-axis).

4.4 Control of the Allen-Cahn equation

Let us consider the following nonlinear Allen-Cahn PDE with homogeneous Neumann boundary conditions:

$$\begin{cases} \partial_t y(t,x) = \sigma \partial_{xx} y(t,x) + y(t,x)(1 - y(t,x)^2) + u(t,x), \\ y(0,x) = y_0(x), \end{cases}$$

with $x \in [0,1]$ and $t \in (0, +\infty)$ and the following cost functional

$$J(u, y_0) = \int_0^\infty \int_0^1 (|y(t,x)|^2 + \gamma |u(t,x)|^2) dx\, dt\,.$$

Approximating the PDE by finite difference schemes with d grid points, we obtain the following ODEs system

$$\dot{y}(s) = A(y)y(s) + u(s),$$

with

$$A(y) = \sigma A_0 + I_d - diag(y \odot y), \quad y \in \mathbb{R}^d,$$

where $\odot$ is the Hadamard product, $I_d \in \mathbb{R}^{d \times d}$ is the identity matrix and A_0 is the tridiagonal matrix arising from the discretization of the Laplacian with Neumann boundary conditions. We fix $\sigma = 10^{-2}$ and $d = 30$. The different techniques will be tested on a set of initial conditions of the form

$$y_0(x) = \sum_{k=1}^{4} \frac{a_k}{2} \cos(2\pi k x) k^{-\beta}, \quad a_k \in \{0, 1\}, \tag{26}$$

where the parameter β is related to the decay of the Fourier coefficients and to the regularity we want to assume. We fix $\beta = 3$. The vector $x = [x_1, \ldots, x_d]$ contains the discretization points of the interval $[0, 1]$.

The supervised learning techniques will be trained over the domain $[-1, 1]^d$, while for the test phase we are going to consider 4 initial conditions in the form (26) varying the vector $a = [a_1, a_2, a_3, a_4]$. To evaluate the feedback law induced by the surrogates of the test functions we integrate the dynamical systems with these controls until a final time $t_{\text{final}} = 60$ and use the trapezoidal rule to estimate the costs along these trajectories. We compare this to the costs induced by the SDRE feedback in err_{cost}.

We start applying the TT Gradient Cross for the approximation of the value function. We fix $n = 6$ Gauss-Legendre nodes per dimension and a stopping tolerance $tol_{stop} = 10^{-4}$. After 17 iterations the algorithm stopped reaching the prescribed tolerance and the final TT rank r is equal to 18. Table 3 reports the errors, the computational cost and the number of samples used during the training phase. We note that for the error $err_{train,2}$ for TT Gradient Cross is below the prescribed tolerance (10^{-4}) and it completes the training of the surrogate model in just almost 20 seconds. The number of training samples is comparable to the complexity of the TT structure which we recall to be equal to $O(dnr^2)$.

The kernel approximation uses the Gaussian kernel with shape parameter $1/\sqrt{d}$, which gave slightly better results than the use of the quadratic Matérn kernel. The neural network uses the same setup and training hyperparameters as in the previous examples. Both the kernel approximation and the neural network are trained using 5000 samples, which were generated randomly in Fourier space. The kernel approximation accuracy is comparable to the TT approximation, while the NN performs worse. In terms of the training time, the kernel methods performs best, while the NN has a long training time due to its iterative stochastic gradient descent optimization. We note that the kernel approximation performs comparably well here compared to the TT, because it is only trained on meaningful inputs which were sampled in Fourier space, while the TT likely provides a suitable approximation of the value function in the whole domain.

Surrogate model	$err_{train,2}$	CPU train (s)	# train samples
TT Gradient Cross	3.48e-5	20.10	121906
NN	2.03e-2	45.80	5000
Kernel	2.147e-5	2.82	5000

Table 3 Comparison in terms of the training phase for the different supervised learning techniques.

Table 4 compares the different surrogate models in the testing phase using 4 initial conditions in the form (26) varying the vector $a \in \mathbb{R}^4$. As regards TT Gradient Cross, the err_{test} reaches the same order of magnitude of the prescribed stopping tolerance for all the studied cases, while for the err_{cost} we loose two orders of magnitude with respect to the err_{test}, due to the fact that the error in the cost takes into account the construction of the control, hence depending on the gradient of the surrogate models.

The kernel approximation and the neural network fail to stabilize the dynamical system in a neighbourhood of the origin: More specifically, when the norm of the trajectories is below a certain threshold denoted by a_{TB}, the error in the approximation of the gradient leads to the synthesis of a non-stabilizing feedback control and the solution is driven far away from the origin. To this end, we consider a modification in the computation of the feedback control already introduced in [43] and denoted as Two-Boxes (TB) approach. More precisely, in the region in which the surrogate is not able to stabilize the system, we substitute the surrogate control with a control given by the Linear Quadratic Regulator, i.e. the solution of the SDRE at the origin. Denoted by P_0 the solution of the Riccati equation corresponding to the Linear Quadratic Regulator, we define the surrogate-TB feedback control as:

$$u^*(x) = \begin{cases} -R^{-1}B^\top Px, & \|x\| \leq a_{TB}, \\ -\frac{1}{2}R^{-1}B(x)^\top \nabla V_{surr}(x), & \|x\| > a_{TB}, \end{cases}$$

where $\nabla V_{surr}(x)$ is the gradient of the surrogate model for the approximation of the value function.

Using this strategy, we note by Table 4 that TT and Kernel-TB achieve almost the same err_{cost}, while NN-TB gets slightly higher results. Regarding the err_{test} we observe that Kernel method achieves the best accuracy with order $O(10^{-6})$, while the NN approach gets better results increasing the number of the terms in the Fourier expansion (26).

	TT Gradient Cross		NN-TB		Kernel-TB	
	err_{test}	err_{cost}	err_{test}	err_{cost}	err_{test}	err_{cost}
$[1,0,0,0]$	1.77e-4	0.0320	1.90e-3	0.0545	1.32e-6	0.0341
$[1,1,0,0]$	2.00e-4	0.0330	2.13e-3	0.0572	8.62e-6	0.0350
$[1,1,1,0]$	2.00e-4	0.0333	8.11e-4	0.0577	1.24e-5	0.0360
$[1,1,1,1]$	1.98e-4	0.0333	8.37e-5	0.0580	1.64e-5	0.0364

Table 4 Comparison in terms of the testing phase with 4 initial conditions in the form (26) for the different supervised learning techniques.

Finally, we show the plots of the uncontrolled and of the controlled solutions of the Allen-Cahn PDE. The top left panel of Figure 5 displays the uncontrolled solution and we can note that at the final time it converges to the stable solution $\overline{y}_1(x) \equiv 1$. In the top right panel and in both lower panels of Figure 5 we can observe the behaviour of the Allen-Cahn solution controlled via the TT Gradient Cross, the Kernel-Two Boxes and the NN-Two Boxes approaches. We note that the control is driving the solution to the unstable equilibrium $\overline{y}_2(x) \equiv 0$ already at the very first time instances, keeping the solution close to $\overline{y}_2(x)$ for the remaining time instances.

5 Conclusions

Overall, supervised learning schemes based on tensor trains, kernel methods and neural networks are able to recover high dimensional functions in an error regime between $O(10^{-6})$ and $O(10^{-3})$ quite reliable. The experiments very nicely show the influence

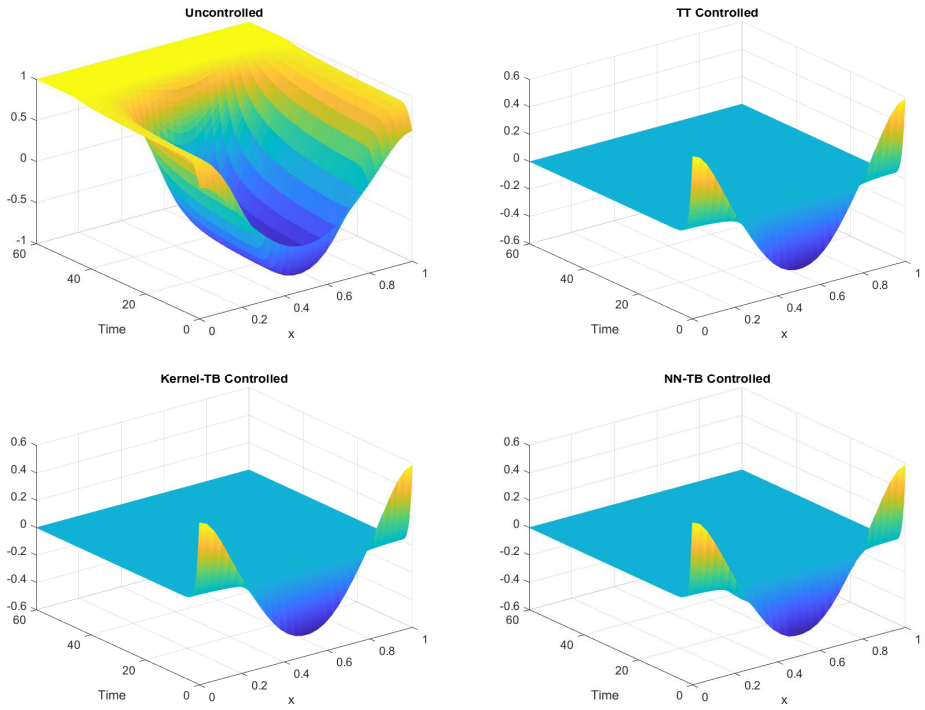

Fig. 5 Solution for the Allen-Cahn equation for the uncontrolled dynamics (top left), the TT Gradient Cross controlled dynamics (top right), the Kernel-Two Boxes controlled dynamics (bottom left) and NN controlled dynamics (bottom right)

of intrinsic structures as regularity and separability (as used in functional tensor train ranks) on the approximation errors. As most of our test cases exhibit some kind of regularity, the TT methods performed very well, as they exploit such assumptions. The kernels models and neural networks could not quite beat tensor trains but allow for a much more general usage: an important advantage of kernel methods and neural networks compared to the TT Cross is especially the ability to use scattered data for the approximation, while TT Cross depends on an active learning mode by adding samples based on a subset of grid points, which makes it difficult to employ on real world data. One can also observe that the metric used to evaluate the functions plays an important role in the assessment of the methods, as the mean squared error used here does not necessarily identify non-differentiabilities or localizations.

The use of available a priori information should be in the centre of high dimensional approximation tasks, when a reasonable high accuracy is required.

Acknowledgements

This manuscript is dedicated to the memory of Maurizio Falcone. Maurizio was not only a great mathematician and an academic father, but also a father figure following all the steps of his academic "sons". He was really interested in high-dimensional

optimal control problems and I hope he will enjoy this paper, whenever he is. Thank you for sharing all your knowledge, your passion for research and your smiles.

Declarations

Ethical Approval Not applicable.

Competing interests The authors have no competing interests.

Authors' contributions All the authors equally contributed to the preparation of the submitted manuscript.

Funding L. Saluzzi was supported by "Gruppo Nazionale per il Calcolo Scientifico" (GNCS - INdAM) and was "titolare di borsa per l'estero dell'Istituto Nazionale di Alta Matematica". M. Oster acknowledges funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - Project number 442047500 through the Collaborative Research Center "Sparsity and Singular Structures" (SFB 1481).

Data Availability Statement A Github repository will be made available upon publication of this work. No data has been used in this paper.

References

[1] Bellman, R.: Dynamic programming. Science **153**(3731), 34–37 (1966)

[2] Bardi, M., Capuzzo-Dolcetta, I.: Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations. Birkäuser, Boston (1997)

[3] Falcone, M., Ferretti, R.: Semi-Lagrangian Approximation Schemes for Linear and Hamilton—Jacobi Equations. SIAM, Philadelphia, PA (2013)

[4] Kalise, D., Kunisch, K.: Polynomial approximation of high-dimensional Hamilton-Jacobi-Bellman equations and applications to feedback control of semilinear parabolic PDEs. SIAM J. Sci. Comput. **40**(2), 629–652 (2018)

[5] Alla, A., Falcone, M., Kalise, D.: An efficient policy iteration algorithm for dynamic programming equations. SIAM Journal on Scientific Computing **37**(1), 181–200 (2015)

[6] Zhao, Z., Yang, Y., Li, H., Liu, D.: Approximate finite-horizon optimal control with policy iteration. In: Proceedings of the 33rd Chinese Control Conference, pp. 8895–8900 (2014)

[7] Tahirovic, A., Astolfi, A.: Optimal control for continuous- time nonlinear systems based on a linear-like policy iteration. In: 2019 IEEE 58th Conference on Decision and Control (CDC), pp. 5238–5243 (2019)

[8] He, S., Fang, H., Zhang, M., Liu, F., Ding, Z.: Adaptive optimal control for a class of nonlinear systems: The online policy iteration approach. IEEE Transactions on Neural Networks and Learning Systems **31**(2), 549–558 (2020)

[9] Luo, B., Wu, H.-N., Huang, T., Liu, D.: Data-based approximate policy iteration for affine nonlinear continuous-time optimal control design. Automatica **50**(12), 3281–3290 (2014)

[10] Pakkhesal, S., Shamaghdari, S.: Sum-of-squares-based policy iteration for sub-optimal control of polynomial time-varying systems. Asian Journal of Control **n/a**

[11] Yazdani, N., Moghaddam, R., Kiumarsi, B., Modares, H.: A safety-certified policy iteration algorithm for control of constrained nonlinear systems. IEEE Control Systems Letters **4**(3), 686–691 (2020)

[12] Tonon, D., Aronna, M., Kalise, D.: Optimal Control: Novel Directions and Applications. Springer, International (2017)

[13] Debrabant, K., Jakobsen, E.: Semi-Lagrangian schemes for linear and fully non-linear Hamilton-Jacobi-Bellman equations. In: Hyperbolic Problems: Theory, Numerics, Applications, pp. 483–490. Springer, International (2014)

[14] Falcone, M.: A numerical approach to the infinite horizon problem of deterministic control theory. Applied Mathematics and Optimization **15**(1), 1–13 (1987)

[15] Falcone, M., Lanucara, P., Seghini, A.: A splitting algorithm for Hamilton-Jacobi-Bellman equations. Applied Numerical Mathematics **15**(2), 207–218 (1994)

[16] Kafash, B., Delavarkhalafi, A., Karbassi, S.M.: Application of variational iteration method for Hamilton-Jacobi-Bellman. Applied Mathematical Modelling **37**(6), 3917–3928 (2013)

[17] Alla, A., Saluzzi, L.: A HJB-POD approach for the control of nonlinear PDEs on a tree structure. Applied Numerical Mathematics **155**, 192–207 (2020)

[18] Akian, M., Gaubert, S., Lakhoua, A.: Convergence analysis of the max-plus finite element method for solving deterministic optimal control problems. In: Proceedings of the IEEE Conference on Decision and Control, pp. 927–934. IEEE, NY (2009)

[19] Akian, M., Fodjo, E.: Probabilistic Max-Plus Schemes for Solving Hamilton-Jacobi-Bellman Equations, pp. 183–209. Springer, International (2018)

[20] Pontryagin, L., Boltyanskii, V., Gamkrelidze, R., Mishchenko, E.: The Mathematical Theory of Optimal Processes. Translated from the Russian by K. N. Trirogoff; edited by L. W.Neustadt. Wiley, New York, NY (1962)

[21] Beeler, S., Tran, H., Banks, H.: Feedback control methodologies for nonlinear systems. Journal of optimization theory and applications **107**(1), 1–33 (2000)

[22] Kang, W., Wilcox, L.: Mitigating the curse of dimensionality: sparse grid characteristics method for optimal feedback control and hjb equations. Computational Optimization and Applications **68**(2), 289–315 (2017)

[23] Nakamura-Zimmerer, T., Gong, Q., Kang, W.: Adaptive deep learning for high-dimensional hamilton–jacobi–bellman equations. SIAM Journal on Scientific Computing **43**(2), 1221–1247 (2021)

[24] Azmi, B., Kalise, D., Kunisch, K.: Optimal feedback law recovery by gradient-augmented sparse polynomial regression. Journal of Machine Learning Research **22**, 1–32 (2021)

[25] Vapnik, V.: Principles of risk minimization for learning theory. In: Advances in Neural Information Processing Systems, pp. 831–838 (1992)

[26] Steinwart, I., Christmann, A.: Support Vector Machines. Springer, Berlin (2008)

[27] Hackbusch, W.: Tensor Spaces And Numerical Tensor Calculus. Springer, Berlin (2012)

[28] Oseledets, I., Tyrtyshnikov, E.: Breaking the curse of dimensionality, or how to use SVD in many dimensions. SIAM J. Sci. Comput. **31**, 3744–3759 (2009)

[29] Oseledets, I.V.: Tensor-train decomposition. SIAM J. Sci. Comput. **33**(5), 2295–2317 (2011)

[30] Khoromskij, B.N.: Tensors-structured numerical methods in scientific computing : survey on recent advances. Chemometrics and intelligent laboratory systems **110**(1), 1–19 (2011)

[31] Hackbusch, W., Schneider, R.: Tensor Spaces and Hierarchical Tensor Representations, pp. 237–261. Springer, Cham (2014)

[32] Bachmayr, M., Schneider, R., Uschmajew, A.: Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations. Found. Comput. Math. **16**(6), 1423–1472 (2016)

[33] Szalay, S., Pfeffer, M., Murg, V., Barcza, G., Verstraete, F., Schneider, R., Legeza: Tensor product methods and entanglement optimization for ab initio quantum chemistry. International j. of quantum chemistry **115**(19), 1342–1391 (2015)

[34] Hackbusch, W.: Numerical tensor calculus. Acta numerica **23**, 651–742 (2014)

[35] Bachmayr, M., Cohen, A., Dahmen, W.: Parametric PDEs: sparse or low-rank approximations? IMA J. of Numerical Analysis **38**(4), 1661–1708 (2017)

[36] Dolgov, S., Kalise, D., Kunisch, K.K.: Tensor Decomposition Methods for High-dimensional Hamilton–Jacobi–Bellman Equations. SIAM Journal on Scientific Computing **43**(3), 1625–1650 (2021)

[37] Oster, M., Sallandt, L., Schneider, R.: Approximating optimal feedback controllers of finite horizon control problems using hierarchical tensor formats. SIAM Journal on Scientific Computing **44**(3), 746–770 (2022)

[38] Oster, M., Sallandt, L., Schneider, R.: Approximating the stationary bellman equation by hierarchical tensor products. Journal of Computational Mathematics (2019)

[39] Stefansson, E., Leong, Y.: Sequential alternating least squares for solving high dimensional linear hamilton-jacobi-bellman equation. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3757–3764 (2016)

[40] Horowitz, A. M.and Damle, Burdick, J.: Linear Hamilton Jacobi Bellman equations in high dimensions. In: 53rd IEEE Conference on Decision and Control, pp. 5880–5887 (2014). IEEE

[41] Fackeldey, K., Oster, M., Sallandt, L., Schneider, R.: Approximative policy iteration for exit time feedback control problems driven by stochastic differential equations using tensor train format. Multiscale Modeling & Simulation **20**(1), 379–403 (2022)

[42] Gorodetsky, A., Karaman, S., Marzouk, Y.: High-dimensional stochastic optimal control using continuous tensor decompositions. The International Journal of Robotics Research **37**(2-3), 340–377 (2018)

[43] Dolgov, S., Kalise, D., Saluzzi, L.: Data-driven tensor train gradient cross approximation for hamilton–jacobi–bellman equations. SIAM Journal on Scientific Computing **45**(5), 2153–2184 (2023)

[44] Götte, M., Schneider, R., Trunschke, P.: A block-sparse tensor train format for sample-efficient high-dimensional polynomial regression. Frontiers in Applied Mathematics and Statistics **7** (2021)

[45] Oseledets, I.V., Tyrtyshnikov, E.E.: TT-cross approximation for multidimensional arrays. Linear Algebra Appl. **432**(1), 70–88 (2010)

[46] Savostyanov, D.V., Oseledets, I.V.: Fast adaptive interpolation of multi-dimensional arrays in tensor train format. In: Proceedings of 7th International Workshop on Multidimensional Systems (nDS). IEEE, NY (2011)

[47] Grasedyck, L., Kriemann, R., Löbbert, C., Nägel, A., Wittum, G., Xylouris, K.: Parallel tensor sampling in the hierarchical Tucker format. Computing and

Visualization in Science **17**(2), 67–78 (2015)

[48] Savostyanov, D.V.: Quasioptimality of maximum–volume cross interpolation of tensors. Linear Algebra Appl. **458**, 217–244 (2014)

[49] Wendland, H.: Scattered Data Approximation. Cambridge Monographs on Applied and Computational Mathematics, vol. 17. Cambridge University Press, Cambridge (2005)

[50] Berner, J., Grohs, P., Kutyniok, G., Petersen, P.: The modern mathematics of deep learning. In: Mathematical Aspects of Deep Learning, pp. 1–111. Cambridge University Press, Cambridge (2022)

[51] DeVore, R.A., Hanin, B., Petrova, G.: Neural network approximation. Acta Numerica **30**, 327–444 (2021)

[52] E, W., Ma, C., Wojtowytsch, S., Wu, L.: Towards a Mathematical Understanding of Neural Network-Based Machine Learning: what we know and what we don't. arXiv (2020)

[53] Higham, C.F., Higham, D.J.: Deep learning: An introduction for applied mathematicians. SIAM Review **61**(4), 860–891 (2019)

[54] Pak, M., Kim, S.: A review of deep learning in image recognition. In: 2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT), pp. 1–3 (2017)

[55] Beck, C., Hutzenthaler, M., Jentzen, A., Kuckuck, B.: An overview on deep learning-based approximation methods for partial differential equations. Discrete and Continuous Dynamical Systems - B **28**(6), 3697–3746 (2023)

[56] Kunisch, Karl, Walter, Daniel: Semiglobal optimal feedback stabilization of autonomous systems via deep neural network approximation. ESAIM: COCV **27**, 16 (2021)

[57] Kunisch, K., Walter, D.: Optimal feedback control of dynamical systems via value-function approximation. arXiv preprint arXiv:2302.13122 (2023)

[58] Darbon, J., Langlois, G.P., Meng, T.: Overcoming the curse of dimensionality for some hamilton–jacobi partial differential equations via neural network architectures. Research in the Mathematical Sciences **7**(3), 1–50 (2020)

[59] Nüsken, N., Richter, L.: Solving high-dimensional hamilton–jacobi–bellman pdes using neural networks: perspectives from the theory of controlled diffusions and measures on path space. Partial Differential Equations and Applications **2**(4), 1–48 (2021)

[60] Ito, K., Reisinger, C., Zhang, Y.: A neural network-based policy iteration algorithm with global h^2 -superlinear convergence for stochastic games on domains. Foundations of Computational Mathematics, 1–44 (2020)

[61] Demo, N., Strazzullo, M., Rozza, G.: An extended physics informed neural network for preliminary analysis of parametric optimal control problems. Computers & Mathematics with Applications **143**, 383–396 (2023)

[62] Han, J., Jentzen, A., E, W.: Solving high-dimensional partial differential equations using deep learning. Proceedings of the National Academy of Sciences **115**(34), 8505–8510 (2018)

[63] Meng, T., Zhang, Z., Darbon, J., Karniadakis, G.E.: SympOCnet: Solving optimal control problems with applications to high-dimensional multi-agent path planning problems. arXiv. OPTdoi: 10.48550/ARXIV.2201.05475 (2022)

[64] Zhou, M., Han, J., Lu, J.: Actor-critic method for high dimensional static hamilton–jacobi–bellman partial differential equations based on neural networks. SIAM Journal on Scientific Computing **43**(6), 4043–4066 (2021)

[65] Onken, D., Nurbekyan, L., Li, X., Fung, S.W., Osher, S., Ruthotto, L.: A neural network approach applied to multi-agent optimal control. In: 2021 European Control Conference (ECC). IEEE, NY (2021)

[66] Ruthotto, L., Osher, S.J., Li, W., Nurbekyan, L., Fung, S.W.: A machine learning framework for solving high-dimensional mean field game and mean field control problems. Proceedings of the National Academy of Sciences **117**(17), 9183–9193 (2020)

[67] Albi, G., Bicego, S., Kalise, D.: Gradient-augmented Supervised Learning of Optimal Feedback Laws Using State-Dependent Riccat Equations. IEEE Control Systems Letters **6**, 836–841 (2022)

[68] Grüne, L.: Computing lyapunov functions using deep neural networks. arXiv preprint arXiv:2005.08965 (2020)

[69] Kunisch, K., Rodrigues, S.S., Walter, D.: Learning an optimal feedback operator semiglobally stabilizing semilinear parabolic equations. Applied Mathematics & Optimization **84**(1), 277–318 (2021)

[70] Kunisch, K., Vásquez-Varas, D., Walter, D.: Learning Optimal Feedback Operators and their Polynomial Approximation. arXiv (2022)

[71] Azmi, B., Kalise, D., Kunisch, K.: Optimal feedback law recovery by gradient-augmented sparse polynomial regression. Journal of Machine Learning Research **22**(48), 1–32 (2021)

[72] Çimen, T.: State-dependent Riccati equation (SDRE) control: a survey. IFAC Proceedings Volumes **41**(2), 3761–3775 (2008)

[73] Alla, A., Kalise, D., Simoncini, V.: State-dependent Riccati equation feedback stabilization for nonlinear PDEs. OPTdoi: 10.48550/ARXIV.2106.07163 (2021)

[74] Banks, H.T., Lewis, B.M., Tran, H.T.: Nonlinear feedback controllers and compensators: a state-dependent Riccati equation approach. Computational Optimization and Applications **37**(2), 177–218 (2007)

[75] Rohrbach, P.B., Dolgov, S., Grasedyck, L., Scheichl, R.: Rank bounds for approximating Gaussian densities in the Tensor-Train format. SIAM/ASA Journal on Uncertainty Quantification **10**(3), 1191–1224 (2022)

[76] Holtz, S., Rohwedder, T., Schneider, R.: The alternating linear scheme for tensor optimization in the tensor train format. SIAM J. Sci. Comput. **34**(2), 683–713 (2012)

[77] Goreinov, S.A., Oseledets, I.V., Savostyanov, D.V., Tyrtyshnikov, E.E., Zamarashkin, N.L.: How to find a good submatrix. In: Olshevsky, V., Tyrtyshnikov, E. (eds.) Matrix Methods: Theory, Algorithms, Applications, pp. 247–256. World Scientific, Hackensack, NY, NY (2010)

[78] Chen, Y., Hosseini, B., Owhadi, H., Stuart, A.M.: Solving and learning nonlinear pdes with gaussian processes. Journal of Computational Physics **447**, 110668 (2021)

[79] Meanti, G., Carratino, L., De Vito, E., Rosasco, L.: Efficient hyperparameter tuning for large scale kernel ridge regression. In: International Conference on Artificial Intelligence and Statistics, pp. 6554–6572 (2022)

[80] Owhadi, H., Yoo, G.R.: Kernel flows: From learning kernels from data into the abyss. Journal of Computational Physics **389**, 22–47 (2019)

[81] Suykens, J.A.: Deep restricted kernel machines using conjugate feature duality. Neural computation **29**(8), 2123–2163 (2017)

[82] Wenzel, T., Marchetti, F., Perracchione, E.: Data-driven kernel designs for optimized greedy schemes: A machine learning perspective. arXiv preprint arXiv:2301.08047 (2023). Accepted for publication in SISC.

[83] Narcowich, F., Ward, J., Wendland, H.: Sobolev bounds on functions with scattered zeros, with applications to radial basis function surface fitting. Mathematics of Computation **74**(250), 743–763 (2005)

[84] Wendland, H., Rieger, C.: Approximate interpolation with applications to selecting smoothing parameters. Numerische Mathematik **101**(4), 729–748 (2005)

[85] Wenzel, T., Santin, G., Haasdonk, B.: Analysis of target data-dependent greedy kernel algorithms: Convergence rates for f-, f· P-and f/P-greedy. Constructive Approximation **57**(1), 45–74 (2023)

[86] Ma, S., Belkin, M.: Kernel machines that adapt to gpus for effective large batch training. Proceedings of Machine Learning and Systems **1**, 360–373 (2019)

[87] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)

[88] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[89] Telgarsky, M.: Representation benefits of deep feedforward networks. arXiv preprint arXiv:1509.08101 (2015)

[90] Jacot, A., Gabriel, F., Hongler, C.: Neural tangent kernel: Convergence and generalization in neural networks. Advances in Neural Information Processing Systems **31** (2018)

[91] Ehring, T., Haasdonk, B.: Hermite kernel surrogates for the value function of high-dimensional nonlinear optimal control problems. arXiv preprint arXiv:2305.06122 (2023)

[92] Dolgov, S., Kalise, D., Saluzzi, L.: Optimizing semilinear representations for state-dependent riccati equation-based feedback control. IFAC-PapersOnLine **55**(30), 510–515 (2022)

[93] Jones, A., Astolfi, A.: On the solution of optimal control problems using parameterized state-dependent Riccati equations. In: 2020 59th IEEE Conference on Decision and Control (CDC), pp. 1098–1103 (2020)