SIAM J. SCI. COMPUT.
Vol. 45, No. 5, pp. A2153–A2184

DATA-DRIVEN TENSOR TRAIN GRADIENT CROSS APPROXIMATION FOR HAMILTON–JACOBI–BELLMAN EQUATIONS*

SERGEY DOLGOV†, DANTE KALISE‡, AND LUCA SALUZZI‡

Abstract. A gradient-enhanced functional tensor train cross approximation method for the resolution of the Hamilton–Jacobi–Bellman (HJB) equations associated with optimal feedback control of nonlinear dynamics is presented. The procedure uses samples of both the solution of the HJB equation and its gradient to obtain a tensor train approximation of the value function. The collection of the data for the algorithm is based on two possible techniques: Pontryagin Maximum Principle and State-Dependent Riccati Equations. Several numerical tests are presented in low and high dimension showing the effectiveness of the proposed method and its robustness with respect to inexact data evaluations, provided by the gradient information. The resulting tensor train approximation paves the way towards fast synthesis of the control signal in real-time applications.

Key words. dynamic programming, optimal feedback control, Hamilton–Jacobi–Bellman equations, tensor decomposition, high-dimensional approximation

MSC codes. 15A69, 15A23, 65F10, 65N22, 49J20, 49LXX, 49MXX

DOI. 10.1137/22M1498401

1. Introduction.

Stabilization of nonlinear dynamical systems is a fundamental problem in control theory, with applications in mechanical systems, chemical engineering, and fluid flow control, among many other areas. Nonlinear stabilization is often approached by means of feedback (closed-loop) controllers which, in contrast to open-loop controls, offer enhanced stability properties with respect to external disturbances. The synthesis of optimal feedback controls resorts to the use of dynamic programming, which characterizes the optimal feedback law in terms of the solution of a Hamilton–Jacobi–Bellman (HJB) nonlinear Partial Differential Equation (PDE). The main drawback of this approach lies in the fact that the HJB equation must be solved on the state space of the dynamical system, often leading to solving a nonlinear PDE in arbitrarily high dimensions. This limitation is referred to as the *curse of dimensionality*, a term coined by Richard Bellman in the '60s and still an active subject of research. Under some specific structural assumptions, as in the case of linear dynamics and a quadratic cost functional, the HJB equation is equivalent to the matrix Algebraic Riccati Equation (ARE), for which high-dimensional solvers are readily available [35, 7]. Unfortunately, for the fully nonlinear setting, no reformulation is possible and the HJB PDE must be solved directly. In this direction, over the last years there has been significant progress toward the solution of high-dimensional HJB PDEs arising in optimal control, including max-plus algebra methods [39, 1, 12],

*Submitted to the journal's Methods and Algorithms for Scientific Computing section May 25, 2022; accepted for publication (in revised form) February 22, 2023; published electronically September 13, 2023.

https://doi.org/10.1137/22M1498401

Funding: This research was supported by Engineering and Physical Sciences Research Council (EPSRC) grants EP/V04771X/1 and EP/T024429/1.

†Department of Mathematical Sciences, University of Bath, North Road, BA2 7AY Bath, United Kingdom (s.dolgov@bath.ac.uk).

‡Department of Mathematics, Imperial College London, South Kensington Campus, SW7 2AZ London, United Kingdom (dkaliseb@imperial.ac.uk, l.saluzzi@imperial.ac.uk).

sparse grids [21], tree-structure algorithms [3, 20], applications of artificial neural networks [29, 13, 38, 40, 58, 43, 50], and regression-type methods in tensor formats [52, 49]. The above-mentioned techniques can scale up to very high-dimensional HJB PDEs; however, the effective implementation of real-time HJB-based controllers remains an open problem.

In this work we develop a data-driven approach based on the knowledge of the value function and its gradient on sample points. Similar ideas have been proposed in [33, 32] in the framework of sparse grids, in [5] with sparse polynomial regression, in [48, 52] via tensor train representation and Monte Carlo quadrature, and in [41, 42, 9, 44] using supervised learning and deep neural networks. The aforementioned works exploit the link between the HJB equation and Pontraygin's Maximum Principle (PMP), a first-order optimality condition which is interpreted as a characteristic curve of the HJB PDE. The latter is used to generate synthetic data for the solution of the HJB equation, whose global solution is then recovered by supervised learning. A similar idea is proposed in [2] where the authors propose a suboptimal feedback law obtained via a feedforward neural network. In this case the training set is generated via the State-Dependent Riccati Equation (SDRE) strategy [6, 11, 4], an extension of the Riccati solution to nonlinear dynamics.

Generally speaking, the proposed methodology belongs to a class of surrogate models, where instead of solving numerically expensive PMP and SDRE problems for each given state of the system, an *offline* approximation of the entire value function is precomputed in a compressed storage format. This format is then used for a cheap *online* evaluation of an approximate control signal for any given state of the system, which allows one to control the system in real time even on a low-performance device (e.g., FPGA). In this paper we propose approximating the value function together with its gradient in a tensor product decomposition, computed via adaptive sampling of either PMP or SDRE. We show that sampling from this prebuilt tensor format of the value function is 100 times faster than the online computation of SDRE solutions, effectively enabling real-time control synthesis.

Tensor decompositions have emerged as efficient approximation techniques for high-dimensional tensors [27, 25], and, when such tensors encapsulate expansion coefficients of functions in a structured basis, multivariate functions [10, 23]. The idea behind tensor decompositions is to approximate a given tensor (or a function) by separation of variables. Convergence of such decompositions for functions of certain regularity [55], or particular classes of value functions [14] has been established. Hierarchical tensor decompositions [27], in particular the tensor train decomposition [45], have become most widely used due to efficient and numerically stable computational algorithms. Those can typically be written in a recursive form that scales linearly with the dimension. The main workhorse is the Alternating Linear Scheme [30, 17], an analogue of the coordinate gradient descent, which optimizes a desired objective function by iterating over the tensor decomposition factors, computing only one factor at a time.

This optimization framework can be used for solving regression problems, such as the Variational Monte Carlo [47, 18, 48, 19]. In this case, one draws random samples from the sought function, and seeks for its tensor approximation by minimizing the misfit on the given samples over the elements of the tensor factors. This problem is also called Tensor Completion [36, 24]. In addition to the straightforward coordinate descent, one can formulate the misfit optimization as a gradient flow on a Riemannian manifold of the tensor decomposition [56], which turns out to be more accurate in a higher-error regime.

Although Tensor Completion works well for smooth functions and rapidly converging decompositions, the predefined sampling may miss localized but significant regions of a more irregular function. Cross approximation methods [46, 54, 26, 53] have been designed to adapt the sampling sets to minimize the conditioning of the interpolation problem, and to improve the approximation accuracy. Moreover, the structure of the sampling sets in the cross methods is aligned to the structure of the sought tensor decomposition, which enables a more efficient linear algebra.

In addition to optimizing the locations of the data samples, one can assimilate more information per each sample. In some problems (such as the optimal control considered here), each evaluation of the sought function comes together with a value of the gradient of this function at little or no extra cost. In this case, one can extend the regression problem such that weighted misfits in both the function and its gradient are minimized. This allows one to take fewer samples for the same accuracy [51, 2], or to achieve a higher accuracy for the same amount of samples. The latter property becomes especially useful when the function values are noisy. In this paper we develop an algorithm to solve the gradient-enhanced regression problem on a tensor train decomposition of the value function of an optimal control problem.

The contributions of this paper can be summarized as follows:

1. We propose a framework for synthetic data generation for infinite horizon nonlinear stabilization problems based on the pointwise solution of SDREs.
2. We formulate a gradient-augmented supervised learning problem where a tensor train approximation of the value function is adaptively trained upon synthetic SDRE samples.
3. We develop a two-box approach (based on the two ingredients above) to improve the accuracy in cases where stabilization towards the origin requires enhanced precision of the control law.
4. We present a comprehensive computational assessment of the proposed methodology over high-dimensional nonlinear tests motivated by optimal control of multiagent systems.

The rest of this paper is structured as follows. In section 2 we give a brief introduction on the infinite horizon optimal control problems and on the two techniques used to generate the dataset: the SDREs and the finite horizon PMP. In section 3 we develop the construction of Gradient Cross for the approximation of the value function. Finally, in section 4 we will demonstrate the efficiency of the proposed algorithm in low- and high-dimensional numerical tests.

2. The infinite horizon optimal control problem and suboptimal feedback laws. In this section we formulate the deterministic infinite horizon problem for which we are interested in synthesizing a feedback law. We first present the optimal feedback synthesis using the HJB formalism to subsequently discuss suboptimal feedback laws which can be effectively cast in a data-driven environment. We consider system dynamics in control affine-form given by

$$\begin{cases} \dot{y}(s) = f(y(s)) + B(y(s))u(s), \ \ s \in (0, +\infty), \\ y(0) = x \in \mathbb{R}^d. \end{cases} \tag{2.1}$$

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 $U \subset \mathbb{R}^m$ is a compact set. We assume that the system dynamics $f : \mathbb{R}^d \to \mathbb{R}^d$ and $B : \mathbb{R}^d \to \mathbb{R}^d$ are $\mathcal{C}^1(\mathbb{R}^d)$ functions, verifying $f(\underline{0}) = B(\underline{0}) = \underline{0}$. Whenever we want to stress the dependence of the control signal from an initial state $x \in \mathbb{R}^d$, we will write $u(t, x)$.

We consider the following undiscounted infinite horizon cost functional:

$$J(u(\cdot,x)) := \int_0^{+\infty} y(s)^\top Q y(s) + u^\top(s) R u(s)\, ds\,, \tag{2.2}$$

where $Q \in \mathbb{R}^{n\times n}$ is a symmetric positive semidefinite matrix and $R \in \mathbb{R}^{m\times m}$ is a symmetric positive definite matrix. Our goal is to synthesize an optimal control in feedback form, that is, a control law that is fully determined upon the current state of the system. We begin by the defining the value function for a given initial condition $x \in \mathbb{R}^d$:

$$V(x) := \inf_{u\in\mathcal{U}} J(u(\cdot,x))\,, \tag{2.3}$$

which, by standard dynamic programming arguments, satisfies the following HJB PDE for every $x \in \mathbb{R}^d$:

$$\min_{u\in U} \left\{ (f(x) + B(x)u)^\top \nabla V(x) + x^\top Q x + u^\top R u \right\} = 0. \tag{2.4}$$

The HJB PDE (2.4) is challenging first-order fully nonlinear PDE cast over $\mathbb{R}^d$, where d can be arbitrarily large, and thus intractable through conventional grid-based methods. However, in the unconstrained case, i.e., $U = \mathbb{R}^m$, the minimizer of the l.h.s. of (2.4) can be computed explicitly as

$$u^*(x) = -\frac{1}{2} R^{-1} B(x)^\top \nabla V(x)\,, \tag{2.5}$$

leading to an unconstrained version of the HJB PDE given by

$$\nabla V(x)^\top f(x) - \frac{1}{4} \nabla V(x)^\top B(x) R^{-1} B(x)^\top \nabla V(x) + x^\top Q x = 0\,. \tag{2.6}$$

In this work, we are interested in recovering an approximation of the optimal feedback law (2.5) circumventing the solution of the high-dimensional HJB PDE (2.6). Instead, we will approximate $V(x)$ in a regression framework, assuming measurements from both the value function $V(x)$ and its gradient $\nabla V(x)$ are available at sampling points. The idea behind this approach resides in the fact that, for a given sample state x, the value function and its gradient can be recovered by minimizing the cost (2.2) without resorting the HJB PDE, and independently from other sampling points. However, generating a single sample of $V(x)$ by solving the associated infinite horizon optimal control problem (2.2) is already a computationally demanding task, unsuitable for the generation of a large-scale dataset. Instead, we propose two alternatives which trade optimality in minimizing (2.2) by fast computability: an SDRE approach and a finite horizon PMP formulation. In this way, we generate a synthetic dataset which approximates the value function and its gradient, leading to a suboptimal, yet asymptotically stabilizing feedback law.

2.1. State-Dependent Riccati Equation. Since the value function is a positive function, with no loss of generality it can be represented as

$$V(x) = x^\top \Pi(x) x, \tag{2.7}$$

where $\Pi(x) \in \mathbb{R}^{n\times n}$ is a symmetric matrix-valued function with its gradient given by the following formula:

$$\nabla V(x) = 2\Pi(x)x + \begin{pmatrix} x^\top \frac{d}{dx_1}\Pi(x) x \\ \vdots \\ x^\top \frac{d}{dx_d}\Pi(x) x \end{pmatrix}. \tag{2.8}$$

In the particular case of a linear dynamics (i.e., $A(x) = A$ and $B(x) = B$), $\Pi(x) = \Pi$ is constant and it is well known that the HJB equation (2.6) becomes the ARE

$$A^\top \Pi + \Pi A - \Pi B R^{-1} B^\top \Pi + Q = 0.$$

An intermediate parametrization can be considered in the nonlinear case, by writing the dynamics in semilinear form

$$\dot{y} = A(y(t))y(t) + B(y(t))u(t). \tag{2.9}$$

In this case, (2.6) can be approximated as

$$A^\top(x)\Pi(x) + \Pi(x)A(x) - \Pi(x)B(x)R^{-1}B(x)^\top \Pi(x) + Q = 0\,, \tag{2.10}$$

which is obtained by applying the ansatz $V(x) = x^\top \Pi(x) x$ with a gradient approximation $\nabla V(x) = 2\Pi(x)x$, that is, by neglecting the second term in (2.8). The resulting equation is known as SDRE. First, we note that the SDRE is a functional equation which must hold for every $x \in \mathbb{R}^d$, hence analytical solutions are only available in limited cases. However, under the assumption that the pair $(A(x), B(x))$ is stabilizable for all $x \in \mathbb{R}^d$, in [6] the authors prove that the feedback law associated with the SDRE

$$u(x) = -R^{-1}B(x)^\top \Pi(x) x \tag{2.11}$$

is locally asymptotically stabilizing (that is, for states in a neighborhood of the origin). Since the SDRE is an approximation of the HJB PDE leading to an optimal feedback law, we claim that the SDRE control is a suboptimal feedback law.

A natural implementation of the SDRE control law for nonlinear stabilization is through a receding horizon approach. In the SDRE setting, this means that given a current state x^k of the trajectory, every matrix in (2.10) is frozen at x^k and an algebraic Riccati equation is solved for $\Pi(x^k)$. Then, the resulting feedback law $u(x^k)$ from (2.11) is applied to evolve the dynamics for a short horizon, until the next state x^{k+1} where the computation is repeated. This implementation is feasible for low-dimensional dynamics, but becomes quickly unpractical as the number of states grows, as it requires the solution of large-scale algebraic Riccati equations at a very high rate. Here instead, we propose a supervised learning approach where (2.10) is used to generate a dataset to approximate $\Pi(x)$ offline, so that online feedback calculations are limited to the evaluation of the feedback law (2.11), which requires the evaluation of $\Pi(x)x$, or the approximate gradient of $V(x)$. The approximation of the value function using a functional tensor train format is discussed in detail in section 2.3.

When approximating the value function via supervised learning, as presented in section 3, we will augment our regression dataset with values of both $V(x)$ and its gradient. This computation relies on the formula (2.8), which requires derivatives of the SDRE solution $\Pi(x)$. Without loss of generality, let us consider the case $B(x) = B$. Computing derivatives of (2.10) with respect to a generic coordinate x_i and denoting by $W = BR^{-1}B^\top$, we obtain the following Lyapunov equation for $\frac{d}{dx_i}\Pi(x)$:

$$\begin{aligned} &\frac{d}{dx_i}\Pi(x)\left(A(x) - W\Pi(x)\right) + \left(A(x)^\top - \Pi(x)W\right)\frac{d}{dx_i}\Pi(x) \\ &\quad = -\frac{d}{dx_i}A(x)^\top \Pi(x) - \Pi(x)\frac{d}{dx_i}A(x), \quad i = 1, \dots, d. \end{aligned} \tag{2.12}$$

Therefore, for each sampled state x, the computation of $V(x)$ and its gradient requires the solution of one ARE (freezing x in (2.10)) and d Lyapunov equations (2.12), where d is the dimension of the dynamical system. If the matrix $A(x)$ does not depend on a variable x_i, its derivative with respect to that variable will be a null matrix and by (2.12) also the matrix $\frac{d}{dx_i}\Pi(x) = 0_d$, will be null. Hence, it will not contribute in the computation of the gradient of the value function. It will be sufficient to solve (2.12) just k times, where k is the number of variables appearing in $A(x)$.

2.2. Pontryagin Maximum Principle. Another option to generate an approximation of the infinite horizon value function is the use of PMP. We refer to Chapter 5.3 in [34] for a complete description of this methodology. However, PMP provides first-order optimality conditions for a finite horizon optimal control problem. Since we are dealing with an infinite horizon, it is necessary to provide a final time T for which the value function and its derivative decay to zero for every initial condition chosen in a reference domain. In this framework, the suboptimality originates from the truncation of the infinite horizon. For an initial condition x and time horizon T, introducing the adjoint variable $p : [0, T] \to \mathbb{R}^d$, the PMP system reads for (2.2)

(2.13)
$$\begin{cases} \frac{d}{dt}y^*(t) = f(y(t)) + B(y^*(t))u^*(t), \\ y^*(0) = x, \\ -\frac{d}{dt}p_i^*(t) = \sum_{j=1}^d p_j^*(t)\partial_{y_i}(f_j(y^*(t)) + B_j(y^*(t))u^*(t)) + 2(Qy^*)_i, \ i = 1, \dots, d, \\ p_i(T) = 0, \\ u^*(t) = -\frac{1}{2}R^{-1}B(y^*(t))^\top p^*(t)\,. \end{cases}$$

In [57] the author shows that the optimal adjoint corresponds to the gradient of the value function along the optimal trajectory. Hence, the value function on the initial condition x will be computed along the optimal trajectory and the optimal control,

$$\begin{aligned} V(x) &= \int_0^{+\infty} y^*(s)^\top Q y^*(s) + u^*(s)^\top R u^*(s)\, ds \\ &\approx \int_0^{\top} y^*(s)^\top Q y^*(s) + u^*(s)^\top R u^*(s)\, ds, \end{aligned} \tag{2.14}$$

while its gradient will be given by initial value of the adjoint, i.e.,

$$\nabla V(x) \approx p(0). \tag{2.15}$$

Equality (2.14) holds with the assumption that the optimal control $u^*(s)$ and the optimal trajectory $y^*(s)$ reached the zero level in the time interval $[0, T]$.

2.2.1. Control constrained problem. Unlike the SDRE approach, using PMP enables the addition of control constraints. For the sake of simplicity, we discuss the scalar control case, i.e., $m = 1$, with a control signal restricted to an interval $[-u_{max}, u_{max}]$. We are going to consider the same approach used in [14]. In order to impose the control constraints, let us choose the following penalty function:

$$\mathcal{P}(u) = u_{\max} \tanh(u/u_{\max}),$$

and let us change the control penalty term $u^\top R u$ in the cost functional (2.2) with the following term:

$$W(u) = 2R \int_0^u \mathcal{P}^{-1}(\mu)\, d\mu. \tag{2.16}$$

Note that the choice of this penalty function will keep the control in the interval $[-u_{max}, u_{max}]$.

The corresponding PMP system becomes

(2.17)
$$\begin{cases} \frac{d}{dt}y^*(t) = f(y(t)) + B(y^*(t))\mathcal{P}(u^*(t)), \\ y^*(0) = x, \\ -\frac{d}{dt}p_i^*(t) = \sum_{j=1}^n p_j^*(t)\partial_{y_j}(f_j(y^*(t)) + B_j(y^*(t))\mathcal{P}(u^*(t))) + 2(Qy^*)_i, \ i = 1, \dots, d, \\ p_i(T) = 0, \\ u^*(t) = -\frac{1}{2}R^{-1}B(y^*(t))^\top p^*(t). \end{cases}$$

In this case we need to compute the value function using the control cost functional (2.16), leading to the formula

$$V(x) = \int_0^\top y^*(s)^\top Q y^*(s) + W(u^*(s))\, ds, \tag{2.18}$$

while the gradient is still obtained by the initial value of the adjoint $p(0)$.

2.3. Functional tensor train. The value function defined by (2.3) lives in the same dimension of the dynamical system (2.1). We are interested in dealing with high-dimensional dynamical systems, e.g., those deriving from the semidiscretization of PDEs or from complex systems. For this reason we need an efficient procedure to deal with high-dimensional functions. We are going to consider the Functional Tensor Train (FTT) to mitigate this problem. We sketch the main aspects, further details can be found in [10, 23].

First, let us fix for each variable x_k a set of n_k basis functions $\Phi_k(x_k) := \{\Phi_k^{(1)}(x_k), \dots, \Phi_k^{(n_k)}(x_k)\}$ and a set of collocation points $X_k = \{x_k^{(i)}\}_{i=1}^{n_k}$. For uniqueness of the representation we assume that the Vandermonde matrix $\Phi_k(X_k) \in \mathbb{R}^{n_k \times n_k}$ is nonsingular. Classical choices for the basis functions are the Lagrange basis or the Legendre polynomials. Now, given a multivariate function $V : X := \bigtimes_{k=1}^d X_k \to \mathbb{R}$, we are interested in the following approximation, which we call FTT:

(2.19)
$$V(x) \approx \tilde{V}(x) := \sum_{\alpha_0=1}^{r_0} \sum_{\alpha_1=1}^{r_1} \cdots \sum_{\alpha_d=1}^{r_d} G^{(1)}_{(\alpha_0,\alpha_1)}(x_1) \cdots G^{(k)}_{(\alpha_{k-1},\alpha_k)}(x_k) \cdots G^{(d)}_{(\alpha_{d-1},\alpha_d)}(x_d).$$

The summation ranges r_k are called *TT ranks* and the factor $G^{(k)}_{(\alpha_{k-1},\alpha_k)}(x_k)$ is called kth *TT core*. Without loss of generality, we can fix $r_0 = r_d = 1$. The TT core is a linear combination of the n_k basis functions:

$$G^{(k)}_{(\alpha_{k-1},\alpha_k)}(x_k) = \sum_{i=1}^{n_k} \Phi_k^{(i)}(x_k) H^{(k)}_{(\alpha_{k-1},i,\alpha_k)} = \Phi_k(x_k) \cdot H^{(k)}_{(\alpha_{k-1},\alpha_k)},$$

where $H^{(k)} \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ is a three-dimensional tensor, $H^{(k)}_{(\alpha_{k-1},i,\alpha_k)} \in \mathbb{R}$ is its element, and $H^{(k)}_{(\alpha_{k-1},\alpha_k)} \in \mathbb{R}^{n_k}$ is a vector from $H^{(k)}$, sliced at the given indices α_{k-1}, α_k. Similarly, we introduce a matrix slice $H^{(k)}_{(i)} \in \mathbb{R}^{r_{k-1} \times r_k}$ and a matrix valued function $G^{(k)}(x_k) : X_k \to \mathbb{R}^{r_{k-1} \times r_k}$. This allows us to ease the notation, and write (2.19) in a matrix form as follows:

$$\tilde{V}(x) = G^{(1)}(x_1) \cdots G^{(k)}(x_k) \cdots G^{(d)}(x_d).$$

In the following sections we are going to refer to the TT rank of a tensor as the maximum among all the TT ranks, $r = \max_{k=0,\dots,d} r_k$.

The TT decomposition was initially written for discrete tensors [45]. A relation to function approximation is established via the same Cartesian basis,

$$\tilde{V}(x) = \sum_{i_1,\dots,i_d=1}^{n_1,\dots,n_d} H_{(i_1,\dots,i_d)} \Phi_1^{(i_1)}(x_1) \cdots \Phi_d^{(i_d)}(x_d).$$

This corresponds to the TT decomposition

$$H_{(i_1,\dots,i_d)} = \sum_{\alpha_0,\dots,\alpha_d=1}^{r_0,\dots,r_d} H^{(1)}_{(\alpha_0,i_1,\alpha_1)} \cdots H^{(d)}_{(\alpha_{d-1},i_d,\alpha_d)}. \tag{2.20}$$

Counting the number of elements in the tensors in the r.h.s., we notice that the TT decomposition needs $\sum_k r_{k-1} n_k r_k = \mathcal{O}(dnr^2)$ degrees of freedom (where we introduce $n := \max_k n_k$), in contrast to $\mathcal{O}(n^d)$ elements in the full tensor of coefficients H. Other tensor decompositions can be used, such as the Hierarchical-Tucker format [28, 27] or the range-separated tensor format [8]. However, in this paper we prefer to use the TT format as the most simple yet general representation, which is suitable for value functions as they are usually smooth.

The TT format admits fast linear algebraic operations. For instance, we can compute multivariate integrals of $\tilde{V}(x)$ by using a tensor product of high-order univariate quadratures. Let $\{x_k^{(i)}\}$ and $\{w_k^{(i)}\}$ be quadrature nodes and weights, respectively. Then the integral $\int V(x)dx$ can be approximated by

$$\left(\cdots \left(\left[\sum_{i_1=1}^{n_1} w_1^{(i_1)} G^{(1)}(x_1^{(i_1)}) \right] \cdot \left[\sum_{i_2=1}^{n_2} w_2^{(i_2)} G^{(2)}(x_2^{(i_2)}) \right] \right) \cdots \right) \cdot \left[\sum_{i_d=1}^{n_d} w_d^{(i_d)} G^{(d)}(x_d^{(i_d)}) \right],$$

requiring $\mathcal{O}(dnr^2)$ operations in this order. Similarly, we can compute derivatives and approximate

$$\nabla_{x_k} V(x) \approx G^{(1)}(x_1) \cdots G^{(k-1)}(x_{k-1}) \cdot \left[\frac{d}{dx_k} G^{(k)}(x_k) \right] \cdot G^{(k+1)}(x_{k+1}) \cdots G^{(d)}(x_d), \tag{2.21}$$

again needing $\mathcal{O}(dnr^2)$ operations per point. Additions, inner and pointwise products, as well as actions of linear operators can be written as explicit TT decompositions with a cost that is linear in d. Such explicit decompositions are likely to have overestimated TT ranks. However, if a tensor (2.20) (or a function (2.19)) is given in a TT format, a quasi-optimal reapproximation can be done in $\mathcal{O}(dnr^3)$ operations by using QR and SVD factorizations. For details, we refer to [45].

Explicit TT formats are a rare exception though. In general, a function may have no exact decomposition, and an approximation must be sought from (as few as possible) evaluations of the function. This can be achieved by solving a Least Squares problem, by minimizing the sum of squares of errors at given (e.g., random) points over the elements of TT cores [47, 18, 48, 19]. However, a priori chosen point sets may miss an important region of the domain, which will make the approximation inaccurate. Alternatively, *TT-Cross* methods [46, 54, 53] adapt the sampling points iteratively towards the optimal locations for the current iterate. However, existing methods employ only the values of the function itself, which may be suboptimal if the values of the gradient are also available for free.

3. Gradient Cross and value function approximation. As discussed in the previous sections, we want to recover the feedback map for the optimal control problem given the knowledge of the value function and its gradient in specific points. In this section we develop a new algorithm in which this information about the value function will enrich the approximation via the FTT. More precisely, given certain sample points $\{x_i\}_{i=1}^N$ and a dataset $\{V(x_i), \nabla V(x_i)\}_{i=1}^N$ computed by either Pontryagin or SDRE, we are interested in determining the coefficient tensors $\{H^{(1)}, \dots, H^{(d)}\}$ which characterize the FTT representation $\tilde{V}(x)$ introduced in (2.19). The regression problem can be formulated as

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

where $\lambda > 0$ is a parameter which weights the contribution of the derivatives, and the gradient of the FTT can be computed considering univariate derivatives (2.21). The resolution of the minimization problem will be addressed in the next sections, first presenting the bivariate case, and then extending the algorithm to higher dimensions.

3.1. Bidimensional Gradient Cross. Given a bidimensional function $V(x_1, x_2)$, we will denote by $V_0(x_1, x_2)$ the function itself and by $V_1(x_1, x_2)$ and $V_2(x_1, x_2)$ its partial derivatives with respect to x_1 and x_2. Let us fix two sets of collocation points $X_1 \in \mathbb{R}^{n_1}$ and $X_2 \in \mathbb{R}^{n_2}$ for each dimension. One may use the matrix $V_i(X_1, X_2) = [V_i(x_1^{(j)}, x_2^{(k)})] \in \mathbb{R}^{n_1 \times n_2}$ for $i \in \{0, 1, 2\}$ to construct a discrete approximation of the function, but these evaluations are expensive, in particular in dimension larger than two.

For this reason we are interested in a TT representation of the form

$$V(x_1, x_2) \approx G^{(1)}(x_1) G^{(2)}(x_2) \tag{3.1}$$

with

$$G^{(1)}(x_1) = \Phi_1(x_1) H^{(1)}, \qquad G^{(2)}(x_2) = H^{(2)} \Phi_2^\top(x_2),$$

where $H^{(1)} \in \mathbb{R}^{n_1 \times r}$, $H^{(2)} \in \mathbb{R}^{r \times n_2}$, and $\{\Phi_k(x)\}_{k=1,2}$ are prefixed basis functions.

More precisely, we are looking for two sets of indices I_1 and I_2 with cardinality $\#I_1 = \#I_2 = r$, and the corresponding interpolating approximations

$$G^{(1)}(x_1) \hat{G}^{(2)}(x_2) \qquad \text{and} \qquad \hat{G}^{(1)}(x_1) G^{(2)}(x_2)$$

such that $G^{(k)}(X_k(I_k)) = \mathtt{I}_r$, where $\mathtt{I}_r$ is the identity matrix, and $\hat{G}^{(1)}(x_1) = V(x_1, X_2(I_2))$, $\hat{G}^{(2)}(x_2) = V(X_1(I_1), x_2)$. This kind of approximation can be obtained via an alternating direction procedure by solving a sequence of least squares problems.

Starting from an initial guess for $H^{(2)}$ and I_2, we want to solve the following problem in the x_1-direction:

$$\min_{H^{(1)}} \sum_{i=0}^{2} \lambda_i \|V_i(X_1, X_2(I_2)) - \tilde{V}_i^1\|^2, \tag{3.2}$$

with

$$\tilde{V}_i^1 = \begin{cases} \Phi_1(X_1) H^{(1)} G^{(2)}(X_2(I_2)), & i = 0, \\ \Phi_1'(X_1) H^{(1)} G^{(2)}(X_2(I_2)), & i = 1, \\ \Phi_1(X_1) H^{(1)} \partial_{x_2} G^{(2)}(X_2(I_2)), & i = 2. \end{cases}$$

In what follows we will fix $\lambda_0 = 1$ and $\lambda_1 = \lambda_2 = \lambda$ are the regularization parameters.

Let us consider for simplicity Lagrangian basis, i.e., $\Phi_k(X_k) = \mathtt{I}_{n_k} \in \mathbb{R}^{n_k \times n_k}$ for $k = 1, 2$. Moreover, we know by hypothesis that $G^{(2)}(X_2(I_2)) = \mathtt{I}_r$. Then, (3.2) is equivalent to the resolution of the following Lyapunov equation:

$$\begin{aligned}(3.3)\qquad &\left(\mathtt{I}_{n_1} + \lambda \Phi_1'(X_1)^\top \Phi_1'(X_1)\right) H^{(1)} + \lambda H^{(1)} \tilde{G}_2 \tilde{G}_2^\top \\ &\quad = V_0(X_1, X_2(I_2)) + \lambda \Phi_1'(X_1)^\top V_1(X_1, X_2(I_2)) + \lambda V_2(X_1, X_2(I_2)) \tilde{G}_2^\top,\end{aligned}$$

where $\tilde{G}_2 = \partial_{x_2} G^{(2)}(X_2(I_2)) = (\Phi_2'(X_2(I_2))\Phi_2(X_2(I_2))^\dagger)^\top$, where $\Phi_2(X_2(I_2))^\dagger = (H^{(2)})^\top$ is the Moore–Penrose pseudoinverse of $\Phi_2(X_2(I_2))$, arising from the condition $G^{(2)}(X_2(I_2)) = \mathtt{I}_r$.

Having solved (3.3), we first execute the QR decomposition of the matrix $H^{(1)} = HR_1$ to improve the numerical stability and then we apply the maximum volume (*maxvol*) method [22] to the matrix H. The *maxvol* algorithm selects the most relevant indices I_1 such that the coefficient matrix $C := HH[I_1, :]^{-1}$ satisfies $\max_{i,j} |C[i, j]| \leq 1 + \delta$, where $\delta > 0$ is an arbitrary threshold. This procedure will provide an approximation of the maximum volume submatrix $H[I_*, :]$, i.e., the submatrix with maximum determinant in modulus among all the possible $r \times r$ submatrices. Taking C as the new $H^{(1)}$ ensures $G^{(1)}(X_1(I_1)) = \mathtt{I}_r$ for the next steps.

Afterwards, we can pass to the least squares problem in the y-direction

$$(3.4)\qquad \min_{H^{(2)}} \sum_{i=0}^{2} \lambda_i \|V_i(X_1(I_1), X_2) - \tilde{V}_i^2\|^2,$$

with

$$\tilde{V}_i^2 = \begin{cases} G^{(1)}(X_1(I_1)) H^{(2)} \Phi_2(X_2)^\top, & i = 0, \\ \partial_{x_1} G^{(1)}(X_1(I_1)) H^{(2)} \Phi_2(X_2)^\top, & i = 1, \\ G^{(1)}(X_1(I_1)) H^{(2)} \Phi_2'(X_2)^\top, & i = 2. \end{cases}$$

After the first step we have $G^{(1)}(X_1(I_1)) = \mathtt{I}_r$ by construction. Then, (3.4) corresponds to the resolution of a Lyapunov equation

$$\begin{aligned}(3.5)\qquad &\lambda \tilde{G}_1^\top \tilde{G}_1 H^{(2)} + H^{(2)} \left(I_{n_2} + \lambda \Phi_2'(X_2)^\top \Phi_2'(X_2)\right) \\ &\quad = V_0(X_1(I_1), X_2) + \lambda \tilde{G}_1^\top V_1(X_1(I_1), X_2) + \lambda V_2(X_1(I_1), X_2) \Phi_2'(X_2),\end{aligned}$$

where $\tilde{G}_1 = \partial_{x_1} G^{(1)}(X_1(I_1))$. The remaining procedure is identical: we solve (3.5), compute the QR decomposition of the solution, and apply the *maxvol* method, obtaining I_2 and $H^{(2)}$. The strategy is repeated until either we converge according to residual criteria or we reach a maximum number of iterations. The method is sketched in Algorithm 3.1.

For the initial guess $H^{(2)}$ one may consider a normally distributed pseudorandom matrix $H^{(2)} \in \mathbb{R}^{r \times n_2}$ and apply steps 7 and 8 of Algorithm 3.1 to obtain the initial set I_2.

With non-Lagrangian bases (i.e., if $\Phi_k(X_k) \neq \mathtt{I}_{n_k}$), we simply need to amend lines 5 and 8 of Algorithm 3.1 to $[I_1, \Phi_1(X_1)H^{(1)}] = maxvol(\Phi_1(X_1)H)$ and $[I_2, \Phi_2(X_2)(H^{(2)})^\top] = maxvol(\Phi_2(X_2)H)$, respectively. This ensures that we select optimal grid points, not optimal coefficients.

The most reliable residual criterion is to compute the mean square approximation error on some validation set (e.g., random points), and to compare it to a chosen threshold. However, a large validation set may inflate the computing time significantly,

Algorithm 3.1 Bidimensional TT Gradient Cross with Lagrangian bases.

1: Choose an initial I_2 and $H^{(2)}$, a tolerance tol and a maximum number of iteration it_{max}
2: **while** $res > tol$ and $it \leq it_{max}$ **do**
3: Solve (3.3) obtaining $H^{(1)}$
4: Compute the QR decomposition $H^{(1)} = HR_1$
5: $[I_1, C] = maxvol(H)$, replace $H^{(1)} = C$.
6: Solve (3.5) obtaining $H^{(2)}$
7: Compute the QR decomposition $(H^{(2)})^\top = HR_1$
8: $[I_2, C] = maxvol(H)$, replace $H^{(2)} = C^\top$.
9: Update res
10: $it = it + 1$
11: **end while**

while a small set may underestimate the error. In the course of existing alternating algorithms [16] it was found that it is sufficient to compare consecutive iterations. Therefore, we proceed with the following definition in line 9 of Algorithm 3.1:

$$res = \max\left\{ \frac{\|H_{it}^{(1)} - H_{it-1}^{(1)}\|_F}{\|H_{it}^{(1)}\|_F}, \frac{\|H_{it}^{(2)} - H_{it-1}^{(2)}\|_F}{\|H_{it}^{(2)}\|_F} \right\},$$

where it is the iteration number.

Once the value function is approximated by Algorithm 3.1, we can compute the optimal control and optimal trajectory starting from a given initial point $x^0 \in \mathbb{R}^2$. The formula for the synthesis of the optimal control is given by

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

where $x = (x_1, x_2) \in \mathbb{R}^2$. The computation of the gradient in this case is simply given by considering the derivatives in (3.1),

$$\partial_{x_1} V(x) = \Phi_1'(x_1) H^{(1)} G_2(x_2),$$
$$\partial_{x_2} V(x) = G_1(x_1) H^{(2)} \Phi_2'^{\top}(x_2).$$

3.2. Multidimensional Gradient Cross. Now we are going to generalize the result obtained in the previous section to an arbitrary dimension d. The FTT representation in this case reads

$$\tilde{V}(x) = G^{(1)}(x_1) \cdots G^{(k)}(x_k) \cdots G^{(d)}(x_d).$$

We will use the alternating strategy in this case, too. For this reason it is convenient to group all the terms before and after the kth TT core, obtaining a more compact formula

$$\tilde{V}(x) = G^{(<k)}(x_{<k}) \cdot G^{(k)}(x_k) \cdot G^{(>k)}(x_{>k}),$$

where

$$G^{(<k)}(x_{<k}) = G^{(1)}(x_1) \cdots G^{(k-1)}(x_{k-1}), \qquad k \geq 2, \tag{3.6}$$
$$G^{(>k)}(x_{>k}) = G^{(k+1)}(x_{k+1}) \cdots G^{(d)}(x_d), \qquad k \leq d-1. \tag{3.7}$$

We are again interested in finding 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 kth step the sets $\overline{X}_{<k}$ and $\overline{X}_{>k}$ are given. Combining the previous expressions, we can write

$$\begin{aligned}\overrightarrow{V}^k &:= \operatorname{vec}\left(V(\overline{X}_{<k}, X_k, \overline{X}_{>k})\right) \approx \tilde{V}^k \\ &:= \left(G^{(<k)}\left(\overline{X}_{<k}\right) \otimes \Phi_k\left(X_k\right) \otimes G^{(>k)}\left(\overline{X}_{>k}\right)\right) \cdot \operatorname{vec}(H^{(k)}),\end{aligned}$$

where $\operatorname{vec}(\cdot)$ stretches a tensor into a vector with the same order of elements, and $\otimes$ is the Kronecker product of matrices. Similar to the bidimensional case, we are going to denote the function values by $\overrightarrow{V}^k_0$ and the derivative values with respect to the ith component by $\overrightarrow{V}^k_i$. Our aim is to solve the following least squares problem:

$$\min_{H^{(k)}} \sum_{i=0}^{d} \lambda_i \| \overrightarrow{V}^k_i - \tilde{V}^k_i \|^2, \tag{3.8}$$

where

$$\tilde{V}^k_i = \begin{cases} \left(G^{(<k)}\left(\overline{X}_{<k}\right) \otimes \Phi_k\left(X_k\right) \otimes G^{(>k)}\left(\overline{X}_{>k}\right)\right) \cdot \operatorname{vec}(H^{(k)}), & i = 0, \\ \left(\partial_i G^{(<k)}\left(\overline{X}_{<k}\right) \otimes \Phi_k\left(X_k\right) \otimes G^{(>k)}\left(\overline{X}_{>k}\right)\right) \cdot \operatorname{vec}(H^{(k)}), & i = 1, \ldots, k-1, \\ \left(G^{(<k)}\left(\overline{X}_{<k}\right) \otimes \Phi'_k\left(X_k\right) \otimes G^{(>k)}\left(\overline{X}_{>k}\right)\right) \cdot \operatorname{vec}(H^{(k)}), & i = k, \\ \left(G^{(<k)}\left(\overline{X}_{<k}\right) \otimes \Phi_k\left(X_k\right) \otimes \partial_i G^{(>k)}\left(\overline{X}_{>k}\right)\right) \cdot \operatorname{vec}(H^{(k)}), & i = k+1, \ldots, d. \end{cases}$$

This least squares problem can be solved as a three-dimensional Sylvester equation as shown in Appendix A.1. Having obtained the solution $H^{(k)}$, we can consider the unfolding matrix $H^{(k)}_L = [H^{(k)}_{(\overline{\alpha_{k-1}, i}, \alpha_k)}] \in \mathbb{R}^{r_{k-1} n_k \times r_k}$ and compute its QR factorization $H^{(k)}_L = \tilde{H}^{(k)}_L R^{(k)}$, then assemble an unfolded TT core

$$\tilde{G}^{(k)}_L = \left(\mathtt{I}_{r_{k-1}} \otimes \Phi_k(X_k)\right) \tilde{H}^{(k)}_L \quad \Leftrightarrow \quad \tilde{G}^{(k)}(x_k) = \sum_{i=1}^{n_k} \Phi^{(i)}_k(x_k) \tilde{H}^{(k)}_{(i)}.$$

Now we can use the *maxvol* method on $\tilde{G}^{(k)}_L$ to find the set I_k, which is a subset of $[\alpha_{k-1}]_{\alpha_{k-1}=1}^{r_{k-1}} \times [i]_{i=1}^{n_k}$. We can split I_k into corresponding components

$$I^{\alpha}_k = \{\alpha_{k-1} : \overline{\alpha_{k-1}, i} \in I_k\}, \quad \text{and} \quad I^{x}_k = \{i : \overline{\alpha_{k-1}, i} \in I_k\}. \tag{3.9}$$

In turn, those enumerate elements in $\overline{X}_{<k}$ and X_k. Therefore, we can define the new interpolation set

$$\overline{X}_{<k+1} := \overline{X}_{<k}(I^{\alpha}_k) \times X_k(I^{x}_k) \tag{3.10}$$

that allows us to continue the iteration. Finally, the new TT core tensor is recovered from the interpolating unfolding matrix $H^{(k)}_L := \tilde{H}^{(k)}_L (\tilde{G}^{(k)}_L[I_k, :])^{-1}$.

Passing on to the $(k+1)$th step, we need to compute $G^{(<k+1)}(\overline{X}_{<k+1})$. Computing the corresponding evaluations of all k cores constituting $G^{(<k+1)}$ will result in an $\mathcal{O}(d^2)$ complexity of the entire algorithm. However, since we can assume that $G^{(<k)}(\overline{X}_{<k})$ was available in the current step, we can obtain $G^{(<k+1)}(\overline{X}_{<k+1})$ with a cost independent of k (and d). These computations are shown in Appendix A.2.

We proceed in the same fashion until either the discrepancy between the consecutive iterations is below the stopping tolerance, or a maximum number of iteration has been reached.

Algorithm 3.2 Gradient TT Cross.

1: Choose initial TT cores $H^{(k)}$, point sets $\overline{X}_{>k}$, a tolerance tol, and a maximum number of iterations it_{max}.
2: **while** $res > tol$ and $it \le it_{max}$ **do**
3: **for** $k = 1, \dots, d$ **do**
4: Find $H^{(k)}$ as the minimizer in (3.8) by solving (A.1).
5: (Optionally) Reduce the TT rank r_k via truncated SVD of $H_L^{(k)}$ to error threshold tol
6: (Optionally) Increase r_k by expanding $H_L^{(k)} := [H_L^{(k)}, Z_L^{(k)}]$ with TT core of error $Z_L^{(k)}$
7: Compute the QR decomposition $H_L^{(k)} = \tilde{H}_L^{(k)} R^{(k)}$
8: $[I_k, (\mathtt{I}_{r_{k-1}} \otimes \Phi_k(X_k)) H_L^{(k)}] = maxvol((\mathtt{I}_{r_{k-1}} \otimes \Phi_k(X_k))\tilde{H}_L^{(k)})$
9: Compute the next point set $\overline{X}_{<k+1}$ as shown in (3.9), (3.10)
10: Compute the next sampled cores as shown in (A.3)
11: Update res
12: **end for**
13: $it = it + 1$
14: **end while**

The method needs a little modification to adapt the TT ranks to a given error threshold. First, if the ranks are overestimated, we can reduce them by computing the singular value decomposition (SVD) instead of the QR decomposition of $H_L^{(k)}$, and truncate the former to ensure that the sum of squares of the truncated singular values is below the desired threshold. Second, if the ranks are underestimated, we can expand $H_L^{(k)}$ with some extra ρ_k columns $Z_L^{(k)}$, thereby increasing the TT rank r_k to $r_k + \rho_k$. It was shown [17] that $Z^{(k)}$ can be taken as TT cores of a TT approximation of the error $z(x) := V(x) - \tilde{V}(x)$. This interplay of the reduction and expansion of the TT ranks will eventually stabilize near optimal ranks for the given error. The entire procedure is summarized in Algorithm 3.2. The expansion cores $Z^{(k)}$ are obtained by running an independent instance of the same algorithm computing a TT approximation $\tilde{z}(x) \approx z(x)$ with fixed TT ranks $\rho_1 = \dots = \rho_{d-1} = \rho$ instead of lines 5 and 6.

3.3. Two Boxes approach. Our scope is to solve the HJB equation (2.4) in a computational domain Ω which is usually a hypercube $[-a, a]^d$ containing all initial conditions of interest and their subsequent optimal trajectories. We discretize the domain separately along each dimension. For the high-dimensional numerical tests we consider Gauss–Legendre nodes and the corresponding Legendre polynomials on the nodes. This choice allow us to obtain accurate solution in the vicinity of the boundary, but it may be less accurate closer to the origin, where the system stabilizes. For this reason we introduce an additional step: the Two Boxes (TB) algorithm. First, we solve the optimal control problem on the whole domain using the TT Gradient Cross, constructing an approximation of the value function V on Ω. Afterwards, we construct the optimal trajectory $\tilde{y}^0(t)$ and the optimal control $\tilde{u}^0(t)$ starting from the origin, $\tilde{y}^0(0) = 0$. The exact path and control are zero constant functions since the origin is an equilibrium of the dynamics, but the approximation may escape from the origin due to approximation errors. In this case we consider the maximum value reached by the dynamics until a final time T. This maximum will be denoted as $\tilde{y}^0_{\max} := \max_t \max_i |\tilde{y}^0_i(t)|$, and we will set $a_{TB} = 2\tilde{y}^0_{\max}$. Afterwards, we construct a

new value function V_{TB} on the subdomain $[-a_{TB}, a_{TB}]^d$. Since the new domain is closer to the origin and smaller than the entire domain, we expect a tensor with smaller TT ranks and evaluations needed in the gradient cross. We will use the information provided by both value functions, defining the optimal feedback map as

$$u^*(x) = \begin{cases} F(\nabla V_{TB}(x)), & \|x\|_\infty \le a_{TB}, \\ F(\nabla V(x)) & \text{otherwise}, \end{cases}$$

with $F(g(x)) = -\frac{1}{2} R^{-1} B(x)^\top g(x)$.

If $\tilde{y}^0_{\max}$ is small enough, the value function V_{TB} would be close to the solution of the Linear Quadratic Regulator (LQR) problem. In the LQR setting the value function reads $V_{LQR} = x^\top \Pi x$, where Π is the solution of the Riccati equation

$$A^\top(0)\Pi + \Pi A(0) - \Pi B(0) R^{-1} B(0)^\top \Pi + Q = 0\,.$$

In this case we can construct the optimal control as

$$u^*(x) = \begin{cases} -R^{-1} B(0)^\top \Pi x, & \|x\|_\infty \le a_{TB}, \\ -\frac{1}{2} R^{-1} B(x)^\top \nabla V(x) & \text{otherwise}. \end{cases}$$

We notice that the second choice provides a faster procedure, since it implies just one resolution of a Riccati equation.

4. Numerical tests. In this section we assess the proposed methodology through different numerical tests. First, we investigate the effect of adding gradient information in the regression in a series of closed-form high-dimensional functions with noisy evaluations. The second numerical test deals with a two- dimensional optimal control problem in which the exact value function is known. We test the efficiency of the method under noise and the effect of introducing control constraints. In the third example we study the three-dimensional Lorenz system. We study the performance of the algorithm reducing the control energy penalty in the cost functional. The last test deals with the Cucker–Smale model, where first we compare the PMP and SDRE approaches for data generation. We study the effect of varying the parameter λ and the selection of an optimal parameter. In the last part of the section, a comparison with a Neural Network approach is presented together with the application of the TB approach. The numerical simulations reported in this paper are performed on a Dell XPS 13 with Intel Core i7, 2.8 GHz, and 16 GB RAM. The codes are written in MATLAB R2021a.

Let us introduce some notations useful for the next sections. We denote by J_T the total discrete cost functional computed by applying directly SDRE to approximate the optimal control problem up to a fixed final time T. The total discrete cost functional computed via TT Gradient Cross up to T will be denoted by $\tilde{J}_T$. Similarly, $\tilde{y}^*(t)$ denotes the discrete trajectory at time t controlled with TT, and $\tilde{u}^*(t)$ denotes the corresponding discrete optimal control at time t.

We define the errors in the computation of the cost function and optimal control as

$$err_J := |J_T - \tilde{J}_T|, \qquad err_u := \sqrt{\sum_{i=0}^{n_t - 1} (t_{i+1} - t_i) |u(t_i) - \tilde{u}(t_i)|^2},$$

respectively, where n_t is the number of time steps t_i produced by the RK4 ODE solver, and the maximum absolute optimal state value at the final time T as

$$\tilde{y}_{\max}(T) := \max_i |\tilde{y}^*_i(T)|.$$

4.1. High-dimensional function approximation. In this first numerical experiment we test the gradient cross algorithm for the approximation of high-dimensional functions. We will study the behavior of the algorithm in presence of different noise levels. We are going to consider the following two functions in dimension d:

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

(b) $f(x) = \exp(-\prod_{i=1}^{d} x_i)$, $x \in [-1,1]^d$.

We fix the dimension $d = 100$, the stopping error threshold for the gradient cross $tol = 10^{-4}$, and we discretize the interval $[-1,1]$ with 33 Legendre–Gauss nodes for each direction. The noise is introduced by adding independent identically distributed normal random numbers with mean 0 and standard deviation σ to the values of the function $f(x)$ and all components of the gradient $\partial_i f(x)$. The error is computed with respect to the adaptive TT-Cross [53] with tolerance 10^{-12} and in absence of noise.

It is easy to see that the function (a) has an exact rank-1 TT decomposition, so we will run the TT-Cross with a fixed rank 1. In Figure 4.1 we show a comparison in terms of the mean approximation error for different λ. The noise magnitude σ varies in the set $\{0\} \cup \{10^{-k}, k = 1, \dots, 6\}$. In absence of noise, the gradient cross with $\lambda = 0$ performs with high precision. Increasing the noise, the higher λ is, the better is the approximation.

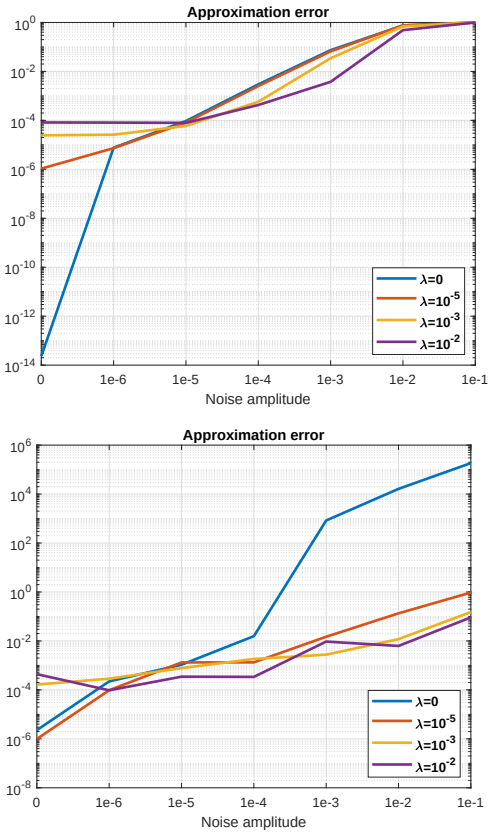

FIG. 4.1. *Mean approximation error for function* (a) *(top) and function* (b) *(bottom) for different λ and noise amplitudes σ. The gradient cross (with $\lambda > 0$) is much more accurate than a gradient-free method ($\lambda = 0$) if a noisy approximate TT approximation is sought.*

Function (b) is a function of rank higher than 1, and it is already possible to notice a different behavior. For every noise amplitude it is possible to find a $\lambda \neq 0$ which obtains a better result compared to the cross approximation without gradient knowledge. In particular, the gradient cross gives a meaningful approximation with an error of 0.1 even with the largest noise magnitude $\sigma = 0.1$, in which case the error of the standard TT-Cross (corresponding to $\lambda = 0$) is larger than 1.

4.2. 2D dynamics with exact solution. The second numerical test deals with an example with exact solution. Given the state dynamics

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ x_1^2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{4.1}$$

and the associated cost functional

$$J = \frac{1}{2} \int_0^\infty \left(\|x(s)\|^2 + |u(s)|^2 \right) ds, \tag{4.2}$$

the solution of the corresponding SDRE is

$$\Pi(x) = \begin{bmatrix} \frac{\sqrt{x_1^4+1}\sqrt{2\sqrt{x_1^4+1}+2x_1^2+1}}{2} & \frac{\sqrt{x_1^4+1}+x_1^2}{2} \\ \frac{\sqrt{x_1^4+1}+x_1^2}{2} & \frac{\sqrt{2\sqrt{x_1^4+1}+2x_1^2+1}}{2} \end{bmatrix}. \tag{4.3}$$

We apply the TT gradient cross described in the previous section and we test it under the effect of noises of different amplitude σ. We discretize the interval $[-1, 1]$ using 14 Lagrangian basis functions and we fix the stopping error threshold for the gradient cross $tol = 10^{-4}$. The collection of the data for the value function and its gradient is performed via the resolution of SDREs. We will consider the case without the knowledge of the gradient (i.e., $\lambda = 0$) and the case with $\lambda = 10^{-4}$. The mean errors in Table 4.1 are computed on a sample of 100 random initial conditions. Since the control is computed taking into account the gradient of the value function, the sum of the considered errors provides an H^1 error estimate. It is possible to notice that in all cases the introduction of the gradient information yields a better approximation. In Figure 4.2 we show the optimal trajectories and the optimal control computed starting from the initial condition $x_0 = (1, -1)$. The right panel of the figure shows the visual coincidence of the three solutions without the presence of noise, which was intuitable by the first row of Table 4.1. The left panel shows the comparison of the solutions under a noise amplitude $\sigma = 10^{-2}$. The choice $\lambda = 0$ and $\lambda = 10^{-4}$ cannot retrieve the starting behavior of the exact control signal, which is zero at the initial time, while fixing $\lambda = 1$ we can observe a better match.

TABLE 4.1
Mean errors in the cost functional of the 2D model and in the control for different amplitudes of noise.

	err_J			err_u		
σ	$\lambda = 0$	$\lambda = 10^{-4}$	$\lambda = 1$	$\lambda = 0$	$\lambda = 10^{-4}$	$\lambda = 1$
0	8.6-9	9.8e-9	2.6e-8	5.0e-7	4.4e-7	1.1e-6
10^{-4}	9.1e-6	6.2e-6	3.4e-6	3.5e-4	1.1e-4	6.4e-5
10^{-3}	6.1e-5	6.2e-5	3.5e-5	2.6e-3	1.7e-3	8.7e-4
10^{-2}	6.8e-4	6.8e-4	3.2e-4	1.2e-2	1.1e-2	5.5e-3
10^{-1}	6.6e-2	2.0e-2	6.2e-3	0.16	0.11	6.0e-2

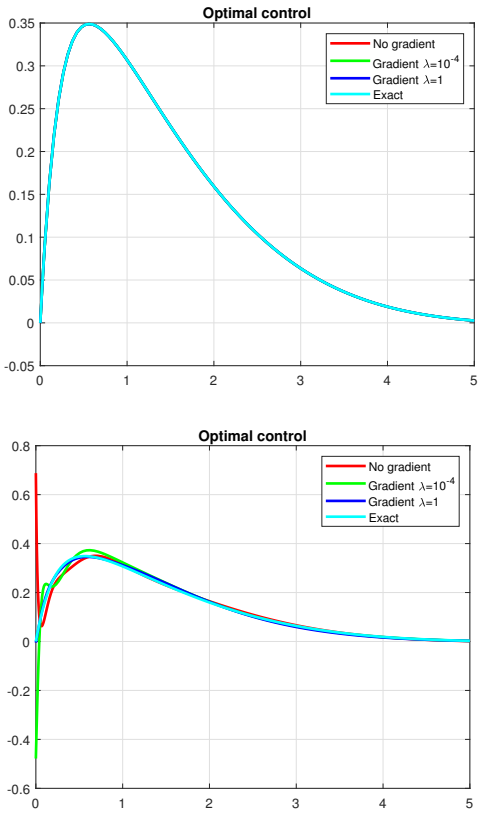

FIG. 4.2. *Optimal control without noise (top) and with noise amplitude* $\sigma = 10^{-2}$ *(bottom) starting from* $x_0 = (1, -1)$.

Constrained control. We focus on the optimal control problem (4.1)–(4.2) coupled with control constraints, i.e., $|u| \leq u_{max}$. As remarked in section 2.2.1, in this case the value function and its gradient will be computed via the PMP, composing the optimal control as

$$u^* = u_{max} \tanh\left(\frac{u}{u_{max}}\right)$$

to enforce the control constraint and the optimization of the cost functional (2.18). We solve the problem in the domain $[-2, 2]^2$ and we consider as initial condition $(x_1(0), x_2(0)) = (2, 2)$. In the first test, we fix the TT-rank $r = 5$. In Table 4.2 we report the values of the total cost obtained using different λ and different constraints. Both in the unconstrained case ($u_{max} = \infty$) and in the constrained cases, it is possible to pick a $\lambda \neq 0$ which performs better than the case with $\lambda = 0$. It is not necessarily the largest λ, as an exceedingly large λ deteriorates the conditioning of the normal equation (A.1). In the left panel of Figure 4.3 we show the optimal trajectories for $\lambda = 0$ in the unconstrained case. In this case the optimal control manages to steer the dynamical system to the equilibrium. However, if we constrain the control to $u_{max} = 20$ (right panel of Figure 4.3), we notice that the control is unable to stabilize the system.

TABLE 4.2

Cost functional $\tilde{J}_T$ for different λ and u_{max} with $r = 5$. In each column there is a nonzero λ that gives the smallest cost.

	$u_{\max} = \infty$	$u_{\max} = 25$	$u_{\max} = 20$
$\lambda = 0$	70.2131	81.9607	93.5168
$\lambda = 10^{-4}$	**70.2124**	81.4676	92.0130
$\lambda = 10^{-3}$	70.2139	81.1542	**91.9481**
$\lambda = 10^{-2}$	70.2134	**81.1451**	92.0824

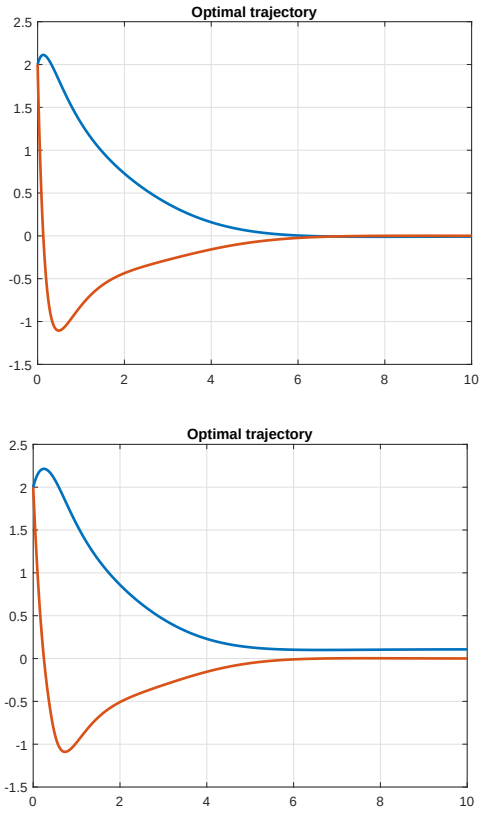

FIG. 4.3. *Optimal trajectory (blue: $x_1(t)$, red: $x_2(t)$) for $u_{max} = \infty$ (top) and $u_{max} = 20$ (bottom) with $\lambda = 0$ and TT rank $r = 5$. This TT rank is too small to approximate the value function accurate enough to stabilize the trajectory. (Figure in color online.)*

This is fixed by increasing the TT-rank of our approximation to $r = 6$. We can see by the right panel of Figure 4.4 that now the solution reaches the origin. In the left panel of Figure 4.4 we show the different behaviors of the control according to the different constraints, fixing $\lambda = 10^{-3}$. Finally, we show in Table 4.3 the total cost for the different choices of λ and $u_{\max}$ with $r = 6$. Reducing the size of the constraint box, the difference between the no-gradient regression and choosing the best λ increases, confirming that the gradient cross achieves a better result for the constrained case in presence of information of both the value function and its gradient.

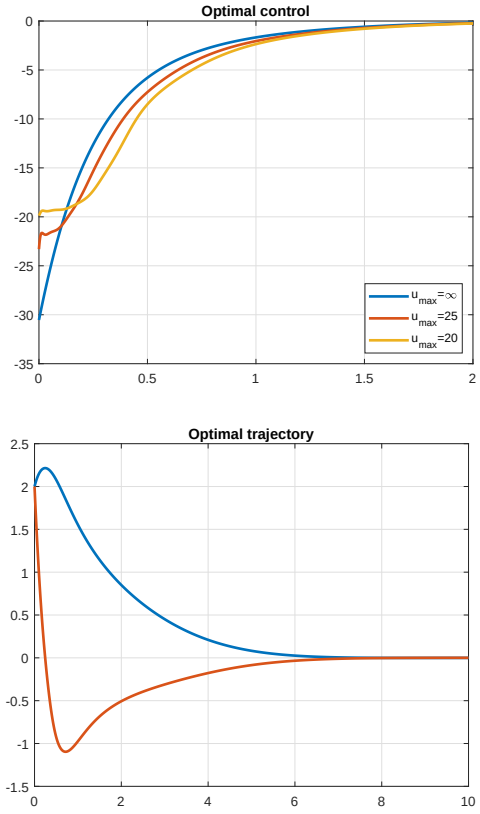

FIG. 4.4. *Optimal control for different u_{max} (top) and optimal trajectory (blue: $x_1(t)$, red: $x_2(t)$) of the 2D model for $u_{max} = 20$ (bottom) with $\lambda = 10^{-3}$ and $r = 6$. This TT rank is sufficient to stabilize the system. (Figure in color online.)*

TABLE 4.3
Cost functional $\tilde{J}_T$ of the 2D model for different λ and u_{max} with $r = 6$.

	$u_{\max} = \infty$	$u_{\max} = 25$	$u_{\max} = 20$
$\lambda = 0$	**70.2123**	81.9629	93.2145
$\lambda = 10^{-4}$	70.2124	81.4544	91.9991
$\lambda = 10^{-3}$	70.2130	81.1660	**91.8594**
$\lambda = 10^{-2}$	70.2131	**81.1548**	92.0458

4.3. Lorenz system. The third example deals with the Lorenz system given by

$$\begin{cases} \dot{x} = \sigma(y - x), \\ \dot{y} = x(\rho - z) - y + u, \\ \dot{z} = xy - \beta z \end{cases} \tag{4.4}$$

with the following cost functional:

$$J = \int_0^\infty \left(|x(s)|^2 + |y(s)|^2 + |z(s)|^2 + \gamma |u(s)|^2 \right) ds. \tag{4.5}$$

The same example has been considered in [37]. We fix $\sigma = 10$, $\beta = 8/3$, $\rho = 2$, and $(x(0), y(0), z(0)) = (-1, -1, -1)$. Data collection is performed via SDRE and we will

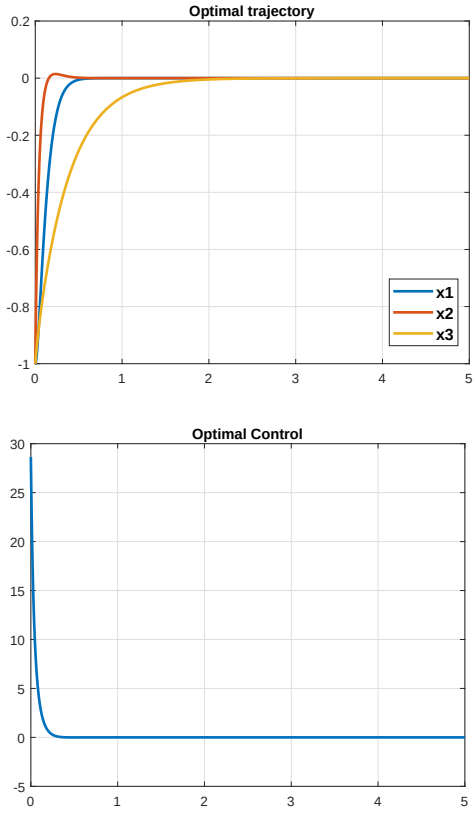

FIG. 4.5. *Optimal trajectory (top) and optimal control (bottom) of the Lorenz model with* $\gamma = 0.001$ *and* $\lambda = 1$.

consider two cases: $\lambda = 0$ and $\lambda = 1$. We will vary the regularization parameter γ in (4.5) in the range $\{0, 0.1, 0.01, 0.001\}$. We consider six Legendre basis functions on the interval $[-1, 1]$ and the stopping error threshold for the gradient cross is 10^{-2}.

In Figure 4.5 we show the optimal trajectory and the optimal control with $\gamma = 0.001$ and $\lambda = 1$. In Figure 4.6 we show the number of time steps needed for `ode45` solver to compute the optimal trajectory in logarithmic scale. It is evident that for small values of the regularization parameter γ, the ODE solver needs more time steps in the no-gradient case compared to the gradient case. The norm of the difference of the two value functions is $1.2 \cdot 10^{-6}$, and the difference of the total cost functionals is $3.5 \cdot 10^{-7}$, which is within the requested error threshold. However, the value function approximation computed with $\lambda = 1$ appears smoother, allowing one to obtain a similar trajectory with fewer time steps.

4.4. Cucker–Smale model. Let us consider the dynamics governed by the Cucker–Smale model with N_a interacting agents given by

$$\begin{bmatrix} \dot{y} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} \mathbb{O}_{N_a} & \mathbb{I}_{N_a} \\ \mathbb{O}_{N_a} & \mathcal{A}_{N_a}(y) \end{bmatrix} \begin{bmatrix} y \\ v \end{bmatrix} + \begin{bmatrix} \mathbb{O}_{N_a} \\ \mathbb{I}_{N_a} \end{bmatrix} u, \tag{4.6}$$

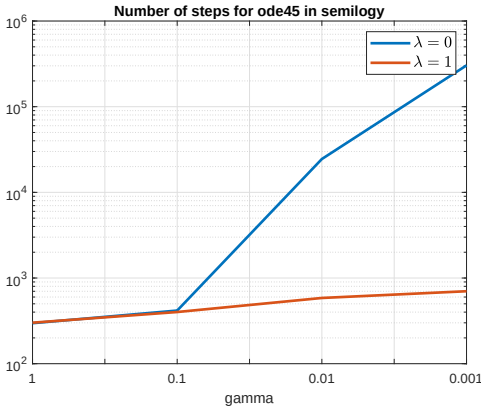

FIG. 4.6. *Numbers of time steps needed for the computation of the controlled trajectory of the Lorenz model using* `ode45` *for different regularization parameters* γ. *For large values of* γ *the two curves overlap, while we note an increasing difference for small* γ. *For the smallest* $\gamma = 10^{-3}$, *the no-gradient method needs three orders of magnitude more time steps compared to the gradient method.*

TABLE 4.4
Comparison between SDRE, PMP, and PMPJ on the Cucker–Smale model with $N_a = 2$. *Here SDRE is the best method.*

	CPU	$\tilde{J}_T$	$\tilde{y}_{\max}(T)$
SDRE	0.5s	0.150168	1.5e-8
PMP	33s	0.150173	2.0e-6
PMPJ	24s	0.150173	6.2e-7

with

$$[\mathcal{A}_{N_a}(y)]_{i,j} = \begin{cases} -\frac{1}{N_a}\sum_{k\neq i} P(y_i, y_k) & \text{if } i = j, \\ \frac{1}{N_a} P(y_i, y_j) & \text{otherwise,} \end{cases}$$

$$P(y_i, y_j) = \frac{1}{1 + \|y_i - y_j\|^2}.$$

Our aim is to minimize the following cost functional:

$$J(y(\cdot), v(\cdot), u(\cdot)) = \frac{1}{N_a}\int_0^\infty \|y(s)\|^2 + \|v(s)\|^2 + \|u(s)\|^2\, ds.$$

We consider a state domain $\Omega = [-0.5, 0.5]^{2N_a}$, 5 Legendre basis functions in each variable, and the gradient cross stopping tolerance is equal to 10^{-2}. We first compare PMP and SDRE to check their performances in producing samples of the cost. In this case we fix $N_a = 2$ and final time $T = 20$ for the corresponding finite horizon control problem for PMP. The PMP system (2.13) is solved via the MATLAB function `bvp4c`. It is possible to supply this function with the Jacobian of the differential equation to accelterate the algorithm and to obtain a more accurate solution. We will denote by PMP the resolution of the system (2.13) without the the knowledge of the Jacobian, while by Pontraygin's Maximum Principle with Jacobian (PMPJ) the one enriched by this further information. The results of the comparison are shown in Table 4.4.

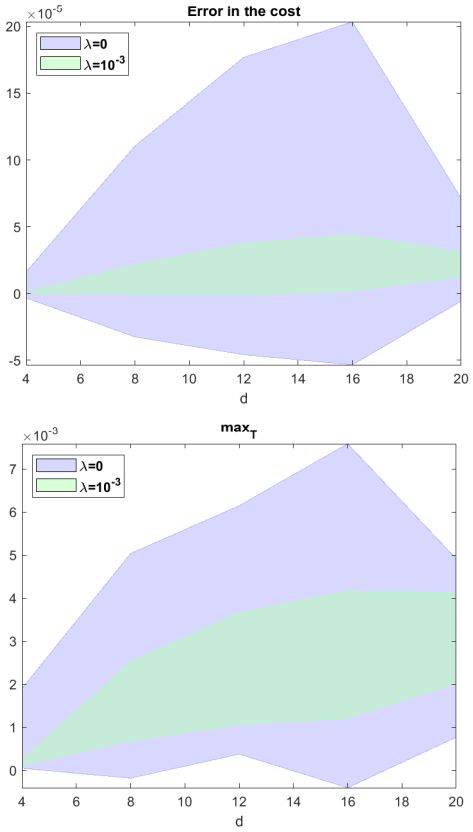

FIG. 4.7. *Error in the Cucker–Smale cost functional (top) and* $\tilde{y}_{\max}(T)$ *(bottom) for* $\lambda = 0$ *(blue) and* $\lambda = 10^{-3}$ *(green). Shaded area denotes mean* $\pm$ 1 *standard deviation over* 10 *runs. Here* $\lambda = 10^{-3}$ *gives a more accurate approximation. (Figure in color online.)*

We can notice that SDRE is 66 times faster than PMP and 48 times faster compared to PMPJ, obtaining almost the same result in terms of the total cost, and a much smaller absolute value of the final state than PMPJ and PMP. In the following run we use the SDRE approach to generate the data for the TT Gradient Cross.

We turn our attention to higher-dimensional problems. We first analyze the behavior of the Gradient Cross algorithm increasing the dimension $d = 2N_a$. In Figure 4.7 we compare the error in the cost functional and the maximum reached by the dynamics at the final time increasing the dimension d from 4 to 20. We consider two cases, one in the absence of gradient information ($\lambda = 0$) and the other one with $\lambda = 10^{-3}$. The shaded areas in the plots are encircled by mean $\pm$ 1 standard deviation over 10 trials. For example, in the case of the error in the cost we consider the mean $\overline{err}_J(d)$ and the corresponding standard deviation $\sigma_{err_J}(d)$. The shaded area is created considering for each dimension the interval $[\overline{err}_J(d) - \sigma_{err_J}(d), \overline{err}_J(d) + \sigma_{err_J}(d)]$. We see that the mean and the standard deviation are both lower in the case with gradient information and this difference grows with increasing dimension of the problem. In Figure 4.8 we show the comparison in terms of TT ranks and evaluations needed by the Gradient Cross. It is important to point out that the TT ranks fluctuate in the same interval [11.5, 16] for all dimensions, proving that we

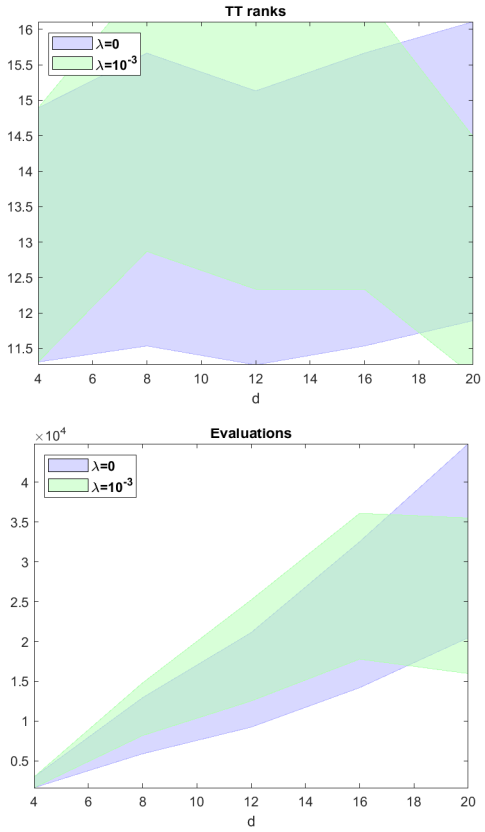

FIG. 4.8. *TT ranks (top) and number of evaluations (bottom) for* $\lambda = 0$ *(blue) and* $\lambda = 10^{-3}$ *(green). Shaded area denotes mean* $\pm$ 1 *standard deviation over* 10 *runs. Both methods show comparable complexity, though* $\lambda = 10^{-3}$ *has less variation from run to run. (Figure in color online.)*

are really solving the curse of dimensionality. In terms of the number of evaluations, for lower dimensions the case with $\lambda = 10^{-3}$ presents a higher mean, but for higher dimensions we obtain a decreasing behavior in contrast to the case with $\lambda = 0$.

The choice of the parameter λ is crucial in this approach. We show a comparison of the performances for different λ and in different dimensions. Since the computational costs of our previous tests are comparable, we will focus instead on the error in the cost functional and on $\tilde{y}_{\max}(T)$. We consider a finite set Λ for the variable λ and we minimize the total cost and the final maximum value on this set. The result of the minimization will provide the best choice for the parameter λ. In Figure 4.9 we report the results for these quantities. The parameter λ is taken from the set $\Lambda = \{0\} \cup \{10^{-k}, k = 0, \dots, 6\}$, and the dimension d varies in the range $\{10, 20, 30, 40\}$. The minimum for each dimension is marked with a circle. First, we can notice that in all the cases the parameter $\lambda = 0$ never represents the optimal choice. We can notice that we obtain a parameter independent on the dimension since in almost all cases $\lambda = 10^{-6}$ represents the optimal choice.

In Table 4.5 we report the averaged elapsed time in computing the suboptimal control applying directly SDRE (first column), via the value function precomputed by

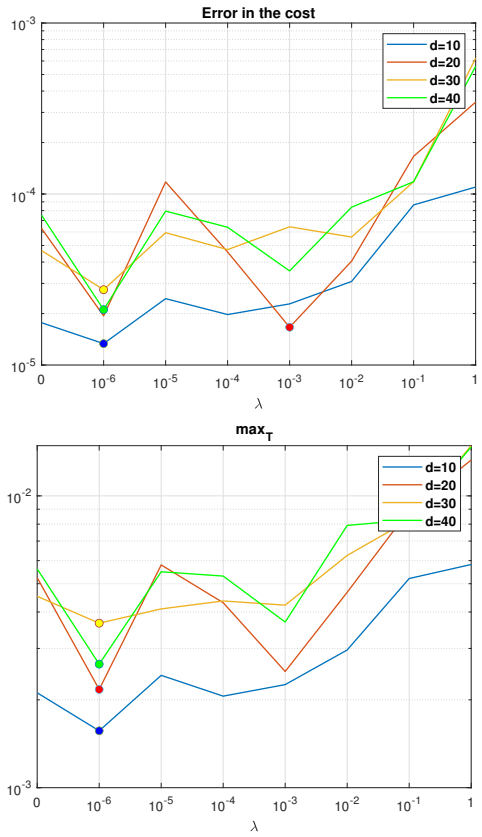

FIG. 4.9. *Error in the Cucker–Smale cost functional (top) and* $\tilde{y}_{\max}(T)$ *(bottom) for different* λ *and dimensions. It is possible to notice that* $\lambda = 10^{-6}$ *represents the optimal choice in almost all cases studied.*

TABLE 4.5
Averaged CPU time for a single computation of the suboptimal control for the different methods.

d	SDRE	TT	Two Boxes
10	$1.4e-3s$	$1.8e-5s$	$1.7e-5s$
20	$5.4e-3s$	$6.9e-5s$	$6.4e-5s$
30	$1.0e-2s$	$1.6e-4s$	$1.4e-4s$
40	$2.2e-2s$	$3.3e-4s$	$1.9e-4s$
100	$1.3e-1s$	$5.3e-3s$	$4.5e-3s$

the TT (second column) and via the value function precomputed by the TB approach (third column). We immediately note that the evaluation of TT is two orders of magnitude faster than the online SDRE solution, proving the efficiency in precomputing the value function when a real-time solution is needed. By comparing the final two columns we notice a small speed-up, showing the beneficial application of LQR in a region close to the origin.

Comparing with neural networks. The aim of this section is to compare the proposed technique with a supervised learning approach discussed in [2]. In this work the authors generate a dataset using an SDRE approach and train a neural network to directly learn a suboptimal feedback map $u(x)$. We will compare against this approach in the framework of agent-based dynamics.

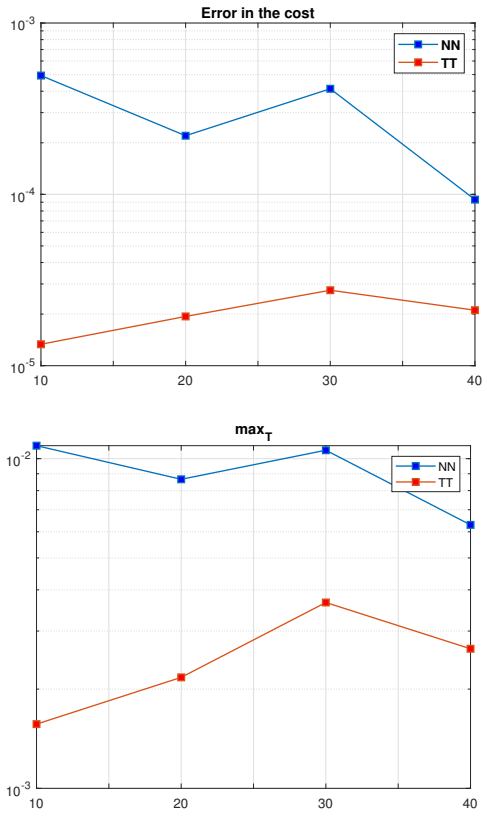

FIG. 4.10. *Error in the cost functional (top) and $\tilde{y}_{\max}(T)$ (bottom) for TT and NN for different dimensions. The TT approximation is more accurate in terms of both indicators for all dimensions.*

In Figure 4.10 we compare the two approaches varying the dimension $d \in \{10, 20, 30, 40\}$. The choice of the parameter λ for TT follows the previous paragraph. For the neural networks (NN) approach we consider the number of samples equal to the number of evaluations needed for TT, in this way the two strategies will have the same computational complexity. We can deduce that the TT approach is up to one order of magnitude more accurate in terms of both indicators.

Now we fix the dimension $d = 40$ and we show the optimal trajectories for the two approaches in Figure 4.11. As noticed in this figure and in the right panel of Figure 4.10, the NN is far from the equilibrium, while TT is evidently close. In Figure 4.12 we show the optimal trajectories starting from $x_0 = \underline{0} \in \mathbb{R}^{40}$ for TT (left panel) and NN (right panel). We note that all the components deviate from the equilibrium. The first N_a components stabilize around a point different from the origin, while the last N_a components return to 0.

To improve the stabilization near the origin, we apply the TB approach to reduce the quantity $\tilde{y}_{\max}(T)$ for both methods. We recall that we are going to consider the subdomain $[-a_{TB}, a_{TB}]^{40}$, where $a_{TB} = 2\tilde{y}^0_{\max}$ with $\tilde{y}^0_{\max}$ computed from Figure 4.12. We set $a_{TB} = 0.009$ for TT and $a_{TB} = 0.0126$ for NN and we apply LQR in the smaller box. The results are shown in Table 4.6. As discussed previously, without the application of the TB algorithm, TT results are more accurate. Coupling the two methods with TB and LQR, we see by the second line of Table 4.6 a remarkable improvement in terms of the final state magnitude. Finally, we show in Figure 4.6 the

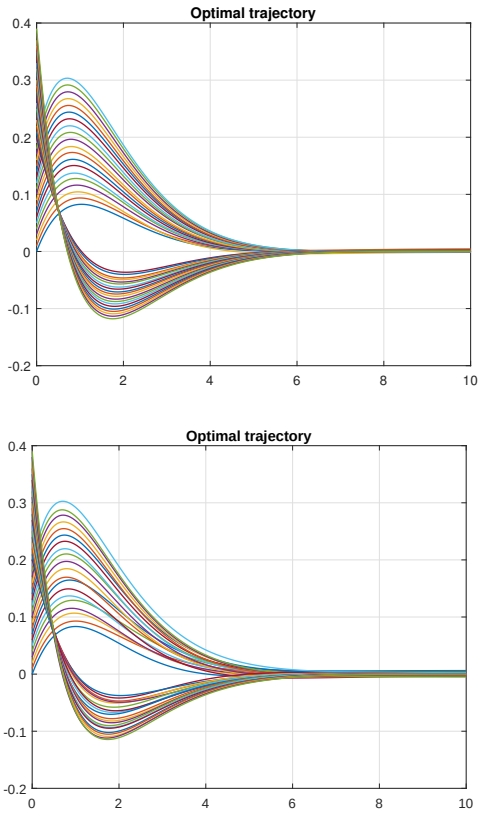

FIG. 4.11. *Optimal trajectory for TT (top) and NN (bottom) with* $d = 40$. *Neither of the methods stabilize the trajectory exactly towards the origin.*

optimal trajectories in this case, where the stabilization to the origin is more visible for both approaches.

Finally, we test the TT approach and the NN strategy fixing the dimension $d = 100$. For the TT cross we fix $\lambda = 10^{-4}$ and the resulting approximation has TT rank equal to 16, in line with the outcome of the left panel of Figure 4.8. Table 4.7 shows the results of the comparison considering both the simple algorithm and the coupling with the TB approach. We set $a_{TB} = 8.6 \cdot 10^{-3}$ for TT and $a_{TB} = 1.2 \cdot 10^{-2}$ for NN, applying LQR in the smaller box. In this case we note that the TT approximation is two orders of magnitude more accurate than NN in both methods in terms of the error in the cost functional. Furthermore, the application of TB is beneficial for $\tilde{y}_{\max}(T)$ for both methods, keeping the error in the cost functional unaffected.

5. Conclusions. We have developed a data-driven method for the approximation of high-dimensional infinite horizon optimal control laws. A key feature of the data-driven methodology is that it circumvents the solution of a HJB PDE, a task that quickly becomes overwhelmingly expensive as the dimension of the state space grows. The value function associated to the feedback law has been written in an FTT form and its approximation has been enriched by the knowledge of both the value function and its gradient at specific sampling points. Synthetic data generation has been performed using two different methods: PMP and the SDRE approach. Through different numerical tests we have shown that the SDRE-based regression per-

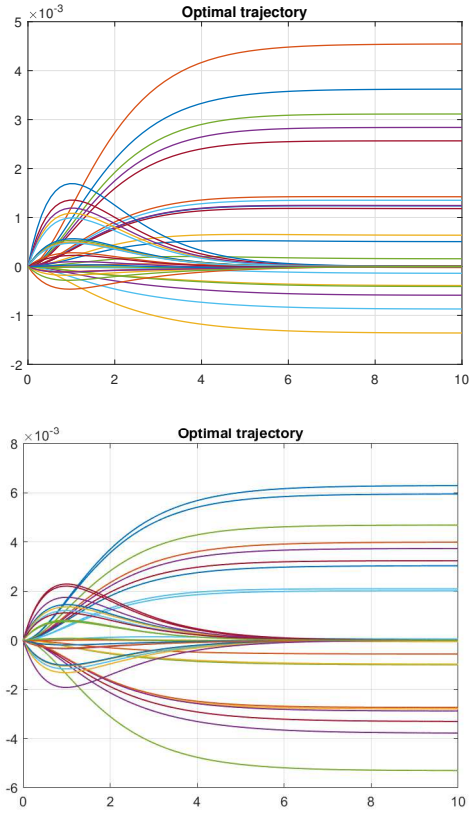

FIG. 4.12. *Optimal trajectory for TT (top) and NN (bottom) with $d = 40$ starting from the origin. The spurious nonzero state value at the final time is used to define the switching threshold in the TB approach.*

TABLE 4.6

Comparison between NN and TT with $\lambda = 10^{-6}$ for $d = 40$. With the TB technique, both methods are more capable to stabilize the state.

Method	NN err_J	TT err_J	NN $\tilde{y}_{\max}(T)$	TT $\tilde{y}_{\max}(T)$
Simple	2.3e-4	2.5e-5	6.2e-3	4.3e-3
TB + LQR	2.0e-4	3.7e-5	4.8e-4	3.3e-4

TABLE 4.7

Comparison between NN and TT with $\lambda = 10^{-4}$ for $d = 100$. The TT approximation achieves an accuracy of two of magnitudes more than NN in terms of error in the cost functional.

Method	NN err_J	TT err_J	NN $\tilde{y}_{\max}(T)$	TT $\tilde{y}_{\max}(T)$
Simple	2.7e-4	2.8e-6	6.0e-3	1.5e-3
TB + LQR	2.7e-4	2.8e-6	4.0e-4	4.2e-4

forms more accurately and efficiently, whereas PMP can still be necessary in the case of state/control constraints. The numerical tests have shown that the introduction of gradient-enhanced supervised learning methodology yields the following advantages with respect to the no-gradient formulation:

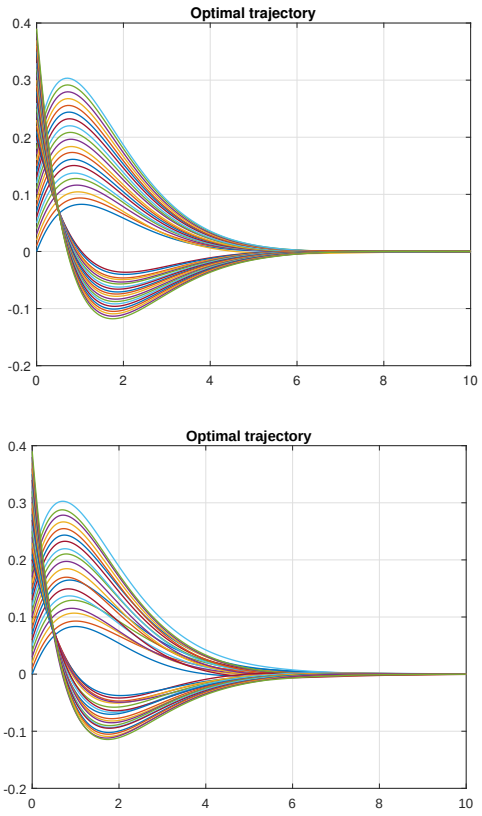

FIG. 4.13. *Optimal trajectory for TB TT (top) and TB NN (bottom) with* $d = 40$. *Both methods are stabilizing.*

- the algorithm presents more stability in presence of noise,
- the H^1 norm of the error is better controlled,
- improved performance for control-constrained cases,
- the feedback map appears more regular and can be integrated with larger time steps,
- it is characterized by a lower standard deviation on the trial set,
- the $Error/Evals$ cost for different θ and dimensions is always minimized picking $\lambda \neq 0$.

We also showed that the maximum TT rank in the representation of the values function grows almost linearly, yielding a effective mitigation of the curse of dimensionality.

In the future we aim at coupling the proposed algorithm with Model Order Reduction techniques in order to deal with problems in considerably higher dimension such as fluid flow control. Since we are not restricted to consider a reduced space in a very low dimension, we are able to work with challenging problems using an extended reduced order basis, leading to a more accurate control design. Further extensions include the study of robust controllers through differential games and Hamilton–Jacobi–Isaacs PDEs as in [31], by resorting to representation via SDREs [4, 15], and data-driven tensor approximation for stochastic control problems in the spirit of [29].

Data access. MATLAB codes implementing the gradient cross and numerical examples are available at https://github.com/saluzzi/TT-Gradient-Cross.

Appendix A. Supplementary materials for the multidimensional Gradient Cross.

A.1. Resolution of the multidimensional least squares problem. We report here the computation for the resolution of the regression problem treated in section 3.2.

Deriving the normal equation for the problem (3.8) we obtain

$$\Big(A_< \otimes M \otimes M_> + M_< \otimes A \otimes M_> + M_< \otimes M \otimes A_>)\mathrm{vec}(H^{(k)}\Big) = \overrightarrow{F}, \tag{A.1}$$

where

$$\begin{aligned}
A_< &= \sum_{i=1}^{k-1} \lambda_i \left(\partial_i G^{(<k)}\left(\overline{X}_{<k}\right)\right)^\top \left(\partial_i G^{(<k)}\left(\overline{X}_{<k}\right)\right),\\
M_< &= G^{(<k)}\left(\overline{X}_{<k}\right)^\top G^{(<k)}\left(\overline{X}_{<k}\right),\\
A &= \lambda_0 \Phi_k\left(X_k\right)^\top \Phi_k\left(X_k\right) + \lambda_k \Phi_k'\left(X_k\right)^\top \Phi_k'\left(X_k\right),\\
M &= \Phi_k\left(X_k\right)^\top \Phi_k\left(X_k\right),\\
A_< &= \sum_{i=k+1}^{d} \lambda_i \left(\partial_i G^{(>k)}\left(\overline{X}_{>k}\right)\right)^\top \left(\partial_i G^{(>k)}\left(\overline{X}_{>k}\right)\right),\\
M_> &= G^{(>k)}\left(\overline{X}_{>k}\right)^\top G^{(>k)}\left(\overline{X}_{>k}\right),
\end{aligned}$$

$$\begin{aligned}
\overrightarrow{F} = &\sum_{i=1}^{k-1} \lambda_i \left[\partial_i G^{(<k)}\left(\overline{X}_{<k}\right)^\top \otimes \Phi_k^\top \otimes G^{(>k)}\left(\overline{X}_{>k}\right)^\top\right] \overrightarrow{V}_i^k\\
&+ \lambda_0 \left[G^{(<k)}\left(\overline{X}_{<k}\right)^\top \otimes \Phi_k^\top \otimes G^{(>k)}\left(\overline{X}_{>k}\right)^\top\right] \overrightarrow{V}_0^k\\
&+ \lambda_k \left[G^{(<k)}\left(\overline{X}_{<k}\right)^\top \otimes \Phi'^\top_k \otimes G^{(>k)}\left(\overline{X}_{>k}\right)^\top\right] \overrightarrow{V}_k^k\\
&+ \sum_{i=k+1}^{d} \lambda_i \left[G^{(<k)}\left(\overline{X}_{<k}\right)^\top \otimes \Phi_k^\top \otimes \partial_i G^{(>k)}\left(\overline{X}_{>k}\right)^\top\right] \overrightarrow{V}_i^k.
\end{aligned}$$

To solve the normal equation (A.1) we first compute a generalized diagonalization for the three couples of Gram matrices $(M_<, A_<)$, (M, A), and $(M_>, A_>)$,

$$M_< V_< = A_< V_< L_<\,, \quad MV = AVL\,, \quad M_> V_> = A_> V_> L_>\,,$$

then (A.1) can be rewritten in the following form:

$$\left(A_< \otimes A \otimes A_>\right)\left(V_< \otimes V \otimes V_>\right) \cdot L_3 \cdot \left(V_<^\top \otimes V^\top \otimes V_>^\top\right)\left(A_< \otimes A \otimes A_>\right)\mathrm{vec}(H^{(k)}) = \overrightarrow{F}, \tag{A.2}$$

where $L_3 = (L_< \otimes L \otimes \mathtt{I} + L_< \otimes \mathtt{I} \otimes L_> + \mathtt{I} \otimes L \otimes L_>)$ is diagonal. Now the linear system (A.2) is easily solvable by inverting individual terms under the Kronecker products, and the diagonal matrix.

A.2. Update of the core evaluations. As stated in section 3.2, in the $(k+1)$th step we can evaluate the core $G^{(<k+1)}(\overline{X}_{<k+1})$ with a cost independent of k and d. Indeed, for each of the r_k elements in I_k^α and I_k^x we compute the matrix products

$$
\begin{aligned}
G^{(<k+1)}\left(\overline{X}_{<k+1}(\alpha_k)\right) &= G^{(<k)}\left(\overline{X}_{<k}(I_k^{\alpha}(\alpha_k))\right) G^{(k)}\left(X_k(I_k^{x}(\alpha_k))\right), \\
\partial_i G^{(<k+1)}\left(\overline{X}_{<k+1}(\alpha_k)\right) &= \partial_i G^{(<k)}\left(\overline{X}_{<k}(I_k^{\alpha}(\alpha_k))\right) G^{(k)}\left(X_k(I_k^{x}(\alpha_k))\right), \\
& \qquad\qquad i = 1, \dots, k-1, \\
\partial_k G^{(<k+1)}\left(\overline{X}_{<k+1}(\alpha_k)\right) &= G^{(<k)}\left(\overline{X}_{<k}(I_k^{\alpha}(\alpha_k))\right) \partial_k G^{(k)}\left(X_k(I_k^{x}(\alpha_k))\right), \\
& \qquad\qquad \alpha_k = 1, \dots, r_k,
\end{aligned} \tag{A.3}
$$

where $G^{(<k)}(\overline{X}_{<k})$ and $\partial_i G^{(<k)}(\overline{X}_{<k})$ are available from the previous step, and need only slicing at $I_k^{\alpha}(\alpha_k)$.

Acknowledgement. For the purpose of open access, the author has applied a Creative Commons Attribution (CC-BY) licence to any Author Accepted Manuscript version arising.

REFERENCES

[1] M. Akian, S. Gaubert, and A. Lakhoua, *The max-plus finite element method for solving deterministic optimal control problems: Basic properties and convergence analysis*, SIAM J. Control Optim., 47 (2008), pp. 817–848, https://doi.org/10.1137/060655286.

[2] G. Albi, S. Bicego, and D. Kalise, *Gradient-augmented supervised learning of optimal feedback laws using state-dependent Riccati equations*, IEEE Control Syst. Lett., 6 (2022), pp. 836–841.

[3] A. Alla, M. Falcone, and L. Saluzzi, *A tree structure algorithm for optimal control problems with state constraints*, Rend. Mat. Appl. (7), 41 (2020), pp. 193–221.

[4] A. Alla, D. Kalise, and V. Simoncini, *State-dependent Riccati equation feedback stabilization for nonlinear PDEs*, Adv. Comput. Math., 49 (2023), 9.

[5] B. Azmi, D. Kalise, and K. Kunisch, *Optimal feedback law recovery by gradient-augmented sparse polynomial regression*, J. Mach. Learn. Res., 22 (2021), 48.

[6] H. T. Banks, B. M. Lewis, and H. T. Tran, *Nonlinear feedback controllers and compensators: A state-dependent Riccati equation approach*, Comput. Optim. Appl., 37 (2007), pp. 177–218.

[7] P. Benner, Z. Bujanovic, P. Kürschner, and J. Saak, *A numerical comparison of different solvers for large-scale, continuous-time algebraic Riccati equations and LQR problems*, SIAM J. Sci. Comput., 42 (2020), pp. A957–A996, https://doi.org/10.1137/18M1220960.

[8] P. Benner, V. Khoromskaia, and B. N. Khoromskij, *Range-separated tensor format for many-particle modeling*, SIAM J. Sci. Comput., 40 (2018), pp. A1034–A1062, https://doi.org/10.1137/16M1098930.

[9] A. Bensoussan, J. Han, S. C. P. Yam, and X. Zhou, *Value-gradient based formulation of optimal control problem and machine learning algorithm*, SIAM J. Numer. Anal., 61 (2023), pp. 973–994, https://doi.org/10.1137/21M1442838.

[10] D. Bigoni, A. P. Engsig-Karup, and Y. M. Marzouk, *Spectral tensor-train decomposition*, SIAM J. Sci. Comput., 38 (2016), pp. A2405–A2439, https://doi.org/10.1137/15M1036919.

[11] J. Cloutier, *State-dependent Riccati equation techniques: An overview*, in Proceedings of the 1997 American Control Conference, Vol. 2, 1997, pp. 932–936.

[12] J. Darbon, P. M. Dower, and T. Meng, *Neural network architectures using min plus algebra for solving certain high dimensional optimal control problems and Hamilton-Jacobi PDEs*, Math. Control Signals Systems 35, (2023), pp. 1–44.

[13] J. Darbon, G. P. Langlois, and T. Meng, *Overcoming the curse of dimensionality for some Hamilton-Jacobi partial differential equations via neural network architectures*, Res. Math. Sci., 7 (2020), 20.

[14] S. Dolgov, D. Kalise, and K. K. Kunisch, *Tensor decomposition methods for high-dimensional Hamilton–Jacobi–Bellman equations*, SIAM J. Sci. Comput., 43 (2021), pp. A1625–A1650, https://doi.org/10.1137/19M1305136.

[15] S. Dolgov, D. Kalise, and L. Saluzzi, *Optimizing semilinear representations for state-dependent Riccati Equation-based feedback control,* IFAC-PapersOnLine, in Proceedings of the 25th International Symposium on Mathematical Theory of Networks and Systems MTNS 2022, 55 (2022), pp. 510–515.

[16] S. V. Dolgov, B. N. Khoromskij, I. V. Oseledets, and D. V. Savostyanov, *Computation of extreme eigenvalues in higher dimensions using block tensor train format*, Comput. Phys. Comm., 185 (2014), pp. 1207–1216.
[17] S. V. Dolgov and D. V. Savostyanov, *Alternating minimal energy methods for linear systems in higher dimensions*, SIAM J. Sci. Comput., 36 (2014), pp. A2248–A2271, https://doi.org/10.1137/140953289.
[18] M. Eigel, R. Schneider, P. Trunschke, and S. Wolf, *Variational Monte Carlo-bridging concepts of machine learning and high-dimensional partial differential equations*, Adv. Comput. Math., 45 (2019), pp. 2503–2532.
[19] K. Fackeldey, M. Oster, L. Sallandt, and R. Schneider, *Approximative policy iteration for exit time feedback control problems driven by stochastic differential equations using tensor train format*, Multiscale Model. Simul., 20 (2022), pp. 379–403, https://doi.org/10.1137/20M1372500.
[20] M. Falcone, G. Kirsten, and L. Saluzzi, *Approximation of optimal control problems for the Navier-Stokes equation via multilinear HJB-POD*, Appl. Math. Comput., 442 (2023), 127722.
[21] J. Garcke and A. Kröner, *Suboptimal feedback control of PDEs by solving HJB equations on adaptive sparse grids*, J. Sci. Comput., 70 (2016), pp. 1–28.
[22] S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, *How to find a good submatrix*, in Matrix Methods: Theory, Algorithms, Applications, V. Olshevsky, and E. Tyrtyshnikov, eds., World Scientific, Hackensack, NY, 2010, pp. 247–256.
[23] A. Gorodetsky, S. Karaman, and Y. Marzouk, *A continuous analogue of the tensor-train decomposition*, Comput. Method Appl. Mech. Eng., 347 (2019), pp. 59–84.
[24] L. Grasedyck and S. Krämer, *Stable ALS approximation in the TT-format for rank-adaptive tensor completion*, Numer. Math., 143 (2019), pp. 855–904.
[25] L. Grasedyck, D. Kressner, and C. Tobler, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitt., 36 (2013), pp. 53–78.
[26] L. Grasedyck, R. Kriemann, C. Löbbert, A. Nägel, G. Wittum, and K. Xylouris, *Parallel tensor sampling in the hierarchical Tucker format*, Comput. Vis. Sci., 17 (2015), pp. 67–78.
[27] W. Hackbusch, *Tensor Spaces and Numerical Tensor Calculus*, Springer-Verlag, Berlin, 2012.
[28] W. Hackbusch and S. Kühn, *A new scheme for the tensor representation*, J. Fourier Anal. Appl., 15 (2009), pp. 706–722.
[29] J. Han, A. Jentzen, and E. W., *Solving high-dimensional partial differential equations using deep learning*, Proc. Natl. Acad. Sci., 115 (2018), pp. 8505–8510.
[30] S. Holtz, T. Rohwedder, and R. Schneider, *The alternating linear scheme for tensor optimization in the tensor train format*, SIAM J. Sci. Comput., 34 (2012), pp. A683–A713, https://doi.org/10.1137/100818893.
[31] D. Kalise, S. Kundu, and K. Kunisch, *Robust feedback control of nonlinear pdes by numerical approximation of high-dimensional hamilton-jacobi–isaacs equations*, SIAM J. Appl. Dyn. Syst., 19 (2020), pp. 1496–1524, https://doi.org/10.1137/19M1262139.
[32] W. Kang and L. Wilcox, *A causality free computational method for HJB equations with application to rigid body satellites*, in Proceedings of the AIAA Guidance, Navigation, and Control Conference, 2015, pp. 1–10.
[33] W. Kang and L. C. Wilcox, *Mitigating the curse of dimensionality: Sparse grid characteristics method for optimal feedback control and HJB equations*, Comput. Optim. Appl., 68 (2017), pp. 289–315.
[34] D. E. Kirk, *Optimal Control Theory*, Dover Publications, Mineola, NY, 2004.
[35] G. Kirsten and V. Simoncini, *Order reduction methods for solving large-scale differential matrix Riccati equations*, SIAM J. Sci. Comput., 42 (2020), pp. A2182–A2205, https://doi.org/10.1137/19M1264217.
[36] D. Kressner, M. Steinlechner, and B. Vandereycken, *Low-rank tensor completion by Riemannian optimization*, BIT Numer. Math., 54 (2014), pp. 447–468.
[37] S. Kundu and K. Kunisch, *Policy iteration for Hamilton–Jacobi–Bellman equations with control constraints*, Comput. Optim. Appl., (2021), in press.
[38] K. Kunisch and D. Walter, *Semiglobal optimal feedback stabilization of autonomous systems via deep neural network approximation*, ESAIM Control, Optim. Calc. Var., 27 (2021), 16.
[39] W. M. McEneaney, *A curse-of-dimensionality-free numerical method for solution of certain HJB PDEs*, SIAM J. Control Optim., 46 (2007), pp. 1239–1276, https://doi.org/10.1137/040610830.

[40] T. Meng, Z. Zhang, J. Darbon, and G. E. Karniadakis, *Sympocnet: Solving optimal control problems with applications to high-dimensional multi-agent path planning problems*, Comput. Methods Sci. Eng., 44 (2022), pp. B1341–B1368, https://doi.org/10.1137/22M1472206.

[41] T. Nakamura-Zimmerer, Q. Gong, and W. Kang, *Adaptive deep learning for high-dimensional Hamilton–Jacobi–Bellman equations*, SIAM J. Sci. Comput., 43 (2021), pp. A1221–A1247, https://doi.org/10.1137/19M1288802.

[42] T. Nakamura-Zimmerer, Q. Gong, and W. Kang, *QRnet: Optimal regulator design with LQR-augmented neural networks*, IEEE Control Syst. Lett., 5 (2021), pp. 1303–1308.

[43] D. Onken, L. Nurbekyan, X. Li, S. W. Fung, S. Osher, and L. Ruthotto, *A neural network approach applied to multi-agent optimal control*, in Proceedings of the 2021 European Control Conference (ECC), IEEE, 2021, pp. 1036–1041.

[44] D. Onken, L. Nurbekyan, X. Li, S. W. Fung, S. Osher, and L. Ruthotto, *A Neural Network Approach for High-dimensional Optimal Control*, preprint, https://arxiv.org/abs/2104.03270, 2021.

[45] I. V. Oseledets, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317, https://doi.org/10.1137/090752286.

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

[47] M. Oster, L. Sallandt, and R. Schneider, *Approximating the Stationary Bellman Equation by Hierarchical Tensor Products*, preprint, https://arxiv.org/abs/1911.00279, 2019.

[48] M. Oster, L. Sallandt, and R. Schneider, *Approximating optimal feedback controllers of finite horizon control problems using hierarchical tensor formats*, SIAM J. Sci. Comput., 44 (2022), pp. B746–B770, https://doi.org/10.1137/21M1412190.

[49] L. Richter, L. Sallandt, and N. Nüsken, *Solving high-dimensional parabolic pdes using the tensor train format*, in International Conference on Machine Learning, PMLR, 2021, pp. 8998–9009.

[50] L. Ruthotto, S. J. Osher, W. Li, L. Nurbekyan, and S. W. Fung, *A machine learning framework for solving high-dimensional mean field game and mean field control problems*, Proc. Natl. Acad. Sci., 117 (2020), pp. 9183–9193.

[51] G. Ryzhakov and I. Oseledets, *Function Approximation Using Gradient Information with Application to Parametric and Stochastic Differential Equations*, preprint, https://arxiv.org/abs/1802.01542, 2018.

[52] L. Sallandt, *Computing High-dimensional Value Functions of Optimal Feedback Control Problems Using the Tensor-Train Format*, Ph.D. thesis, Technische Universität Berlin, Berlin, Germany, 2022.

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

[54] D. V. Savostyanov and I. V. Oseledets, *Fast adaptive interpolation of multi-dimensional arrays in tensor train format*, in Proceedings of 7th International Workshop on Multidimensional Systems (nDS), IEEE, 2011, pp. 1–8.

[55] R. Schneider and A. Uschmajew, *Approximation rates for the hierarchical tensor format in periodic Sobolev spaces*, J. Complexity, 30 (2014), pp. 56–71.

[56] M. Steinlechner, *Riemannian optimization for high-dimensional tensor completion*, SIAM J. Sci. Comput., 38 (2016), pp. S461–S484, https://doi.org/10.1137/15M1010506.

[57] N. Subbotina, *The method of characteristics for the Hamilton-Jacobi equations and its applications in dynamic optimization*, J. Math. Sci., 135 (2006), pp. 2955–3091.

[58] M. Zhou, J. Han, and J. Lu, *Actor-critic method for high dimensional static Hamilton–Jacobi–Bellman partial differential equations based on neural networks*, SIAM J. Sci. Comput., 43 (2021), pp. A4043–A4066, https://doi.org/10.1137/21M1402303.