Journal of Scientific Computing (2024) 101:41
https://doi.org/10.1007/s10915-024-02683-2

A Multilinear HJB-POD Method for the Optimal Control of PDEs on a Tree Structure

Gerhard Kirsten[1] · Luca Saluzzi[2]

Received: 12 May 2023 / Accepted: 17 September 2024 / Published online: 2 October 2024

Abstract
Optimal control problems driven by evolutionary partial differential equations arise in many industrial applications and their numerical solution is known to be a challenging problem. One approach to obtain an optimal feedback control is via the Dynamic Programming principle. Nevertheless, despite many theoretical results, this method has been applied only to very special cases since it suffers from the *curse of dimensionality*. Our goal is to mitigate this crucial obstruction developing a version of dynamic programming algorithms based on a tree structure and exploiting the compact representation of the dynamical systems based on tensors notations via a model reduction approach. Here, we want to show how this algorithm can be constructed for general nonlinear control problems and to illustrate its performances on a number of challenging numerical tests introducing novel pruning strategies that improve the efficacy of the method. Our numerical results indicate a large decrease in memory requirements, as well as computational time, for the proposed problems. Moreover, we prove the convergence of the algorithm and give some hints on its implementation.

Keywords Dynamic programming · Optimal control · Tree structure · Model order reduction · Error estimates · Tree structure algorithm

Mathematics Subject Classification 49L20 · 49J15 · 49J20 · 93B52

1 Introduction

Feedback control is a fundamental concept in engineering and applied mathematics, where the goal is to design a system that can regulate a process to achieve a desired behavior. One of

In loving memory of Maurizio Falcone.

✉ Luca Saluzzi
luca.saluzzi@sns.it

Gerhard Kirsten
gpkirsten@gmail.com

1 Dipartimento di Matematica, Università di Bologna, Piazza di Porta S. Donato, 5, 40127 Bologna, Italy

2 Centro di Ricerche Matematiche "Ennio De Giorgi", Scuola Normale Superiore, Piazza dei Cavalieri, 3, 56126 Pisa, Italy

the most powerful tools in feedback control is the Hamilton Jacobi Bellman (HJB) equation, which provides a framework for optimal control of dynamical systems. The HJB equation is a Partial Differential Equation (PDE) that arises from the calculus of variations and has a wide range of applications, including e.g. robotics, aerospace and finance. The main disadvantage of this approach comes in the form of the so-called *curse of dimensionality*; the phenomenon for which the complexity of a problem increases exponentially as the number of variables or dimensions involved in the problem grows. In real applications, the dynamical system may be described by a large number of state variables either because the continuous problem is in high-dimension or because the dynamical system is obtained via a discretization in space of a PDE. In the context of linear dynamics and quadratic cost functional, the HJB is equivalent to the Differential Riccati Equation for the finite horizon control problem and to the Algebraic Riccati Equation in the infinite horizon case. This setting has been widely researched, leading to several promising high-dimensional solvers [12, 31].

For general nonlinear problems, a Riccati-like reformulation does not exist and the HJB equation must be tackled directly. In the recent years several efforts have been employed in the mitigation of the curse of dimensionality arising in optimal control, among those we mention sparse grids [24], max-plus algebra [1, 2, 15, 37], artificial neural networks [16, 26, 35, 38, 39, 46, 56], the application of tensor formats [19, 40, 45] and radial basis functions [6].

In this paper we aim to mitigate the curse of dimensionality via a graph-based optimization algorithm, the Tree Structure Algorithm (TSA) for the resolution of the finite horizon HJB problem [3]. The TSA leads to the construction of a tree in the direction of all the possible controlled trajectories. Due to its flexible structure, this technique has been already applied in different settings, e.g. high-order schemes [4] and high-dimensional semidiscrete PDEs [7] and its convergence is ensured by rigorous error estimates [48]. Furthermore, a *geometrical pruning* based on the distance of the nodes has been introduced to avoid the exponential growth of the tree and obtain a quadratic growth rate in the context of LQR problems [48]. Unfortunately, for general nonlinear problems this criterion may be not effective since the tree nodes may spread out faster, leading to a further curse of dimensionality. This is the first shortcoming of the TSA that we will aim to address in this paper. We will investigate:

- a bilinear setting where the application of the geometrical pruning also yields a good reduction in the cardinality of the tree,
- an optimal control problem based on monotone controls with an efficient tree-based data structure,
- a *statistical pruning* rule based on the iterative knowledge of the value function on the tree nodes.

The main novelty of the paper lies in the introduction of these innovative pruning strategies that significantly augment the effectiveness of the method.

A second shortcoming of the TSA that we address in this paper is related to the computational cost of evaluating and constructing full-dimensional tree nodes. Given the exponential growth in the cardinality of the tree, that can merely be mitigated by pruning techniques, a massive computational effort may be required to construct and evaluate the discrete problem on the tree nodes, as the dimension of the discrete problem is increased.

A first attempt to address this issue was proposed in [7] where the authors applied a combination of the Proper Orthogonal Decomposition (POD) [55] for the linear part of the problem and Discrete Empirical Interpolation method (DEIM) [13] for the nonlinear terms, to reduce the complexity of the problem. The coupling of the POD technique and the HJB equation dates back to the pioneering paper by Kunisch and co-authors [34], which was then further developed in a series of works [27, 33, 36]. Nevertheless, in this setting, the POD-DEIM algorithm itself has some computational drawbacks. More precisely, the dimension of the vectors that need to be stored for constructing the tree and the POD and DEIM spaces increase exponentially to the order of the dimension of the underlying dynamical system. This may lead to a large bottleneck in the computation of the reduced spaces for larger problems.

Instead, in this paper, we take advantage of the composite structure of a subset of semilinear PDEs, where the underlying PDEs can be written and discretized in Array form, leading to discrete semilinear Matrix and Tensor equations [17, 29, 41, 49]. Given this particular structure, we aim to show how the tree can be constructed in low-dimension by applying a Higher-Order POD-DEIM (HO-POD-DEIM) model order reduction [29] to the discrete problem and then solving the HJB equation on the low-dimensional tree. Our computational results on several benchmark problems indicate that the new algorithm leads to memory requirements that increase linearly in the dimension of the underlying dynamical system, instead of exponentially. Furthermore, the convergence of the proposed technique is established by the derivation of rigorous error estimates for the reduced discrete dynamical system and for the discrete value function computed on the reduced tree. These theoretical results extend the error bounds obtained in [52] to semi-implicit schemes as well as the proposed HO-POD-DEIM technique.

Our construction focuses on general high-dimensional semidiscretized PDEs. A simplified matrix-oriented version of our framework is experimentally explored for the Navier–Stokes (NS) equation in the companion manuscript [22], where the application to systems of differential equations is also discussed. In the previous work, the method was developed for a two-dimensional NS equation. In contrast, the current paper addresses general d-dimensional PDEs. Additionally, we introduce a novel snapshot selection technique and an efficient memory allocation strategy, which reduces memory requirements during the offline phase. The NS equation is a well-known example recognized for its computational expense, where the application of MOR techniques helps in the computation of the solution (see e.g. [43, 44, 53]). Here we consider complex problems in high dimension, showing the promising numerical results on benchmark problems. Furthermore, we deepen the analysis of all the ingredients of this new methodology, including pruning techniques, error estimates and important implementational nuances. We believe that the numerical simulations presented in the last section illustrate that DP is now also feasible for more complex, higher-dimensional problems from a computational point of view, and we hope that this brings it closer to the application of challenging industrial problems.

The paper is organized as follows. In the second section we introduce the optimal control framework and the Tree Structure Algorithm. Section 3 is devoted to the Model Order Reduction setting and its coupling with the TSA, whereas in Sect. 4 we present some hints for an efficient implementation of the proposed algorithm. In Sect. 5 we examine different pruning criteria for the TSA showing some results in the reduction of the cardinality of the tree, and Sect. 6 presents an error bound for the approximation of the value function via the reduced order model algorithm. Finally, in the last section we present some numerical experiments to show the effectiveness of the proposed method.

2 The Optimal Control Problem

Let us consider the classical *finite horizon optimal control problem* that we use as a model problem. The system is driven by

$$\begin{cases} \dot{y}(s) = f(y(s), u(s), s), & s \in (t, T], \\ y(t) = x \in \mathbb{R}^N. \end{cases} \tag{2.1}$$

Here, $y : [t, T] \to \mathbb{R}^N$ is the solution, $u : [t, T] \to \mathbb{R}^m$ is the control, $f : \mathbb{R}^N \times \mathbb{R}^m \times [t, T] \to \mathbb{R}^N$ is the dynamics and

$$\mathcal{U} = \{u : [t, T] \to U, \text{measurable}\}$$

is the set of admissible controls where $U \subset \mathbb{R}^m$ is a compact set. We define the cost functional for the finite horizon optimal control problem as

$$J_{x,t}(u) := \int_t^T L(y(s, u), u(s), s)\, ds + g(y(T)), \tag{2.2}$$

where $L : \mathbb{R}^N \times \mathbb{R}^m \times [t, T] \to \mathbb{R}$ is the running cost and $g : \mathbb{R}^N \to \mathbb{R}$ is the final cost. In the present work we will assume that the functions f, L and g are bounded:

$$\begin{aligned} &|f(x, u, s)| \le M_f, \quad |L(x, u, s)| \le M_L, \quad |g(x)| \le M_g, \\ &\qquad\qquad \forall x \in \mathbb{R}^N, u \in U \subset \mathbb{R}^m, s \in [t, T], \end{aligned} \tag{2.3}$$

the functions f and L are Lipschitz-continuous with respect to the first variable

$$\begin{aligned} &|f(x, u, s) - f(y, u, s)| \le L_f|x - y|, \quad |L(x, u, s) - L(y, u, s)| \le L_L|x - y|, \\ &\qquad \forall x, y \in \mathbb{R}^N, u \in U \subset \mathbb{R}^m, s \in [t, T], \end{aligned} \tag{2.4}$$

and finally the cost g is also Lipschitz-continuous:

$$|g(x) - g(y)| \le L_g|x - y|, \quad \forall x, y \in \mathbb{R}^N. \tag{2.5}$$

Note that these assumptions guarantee uniqueness for the trajectory $y(t)$ by the Carathéodory theorem (we refer to e.g. [10] for a precise statement).

The aim is to construct a state-feedback control law $u(t) = \Phi(y(t), t)$, in terms of the state equation $y(t)$, where Φ is the feedback map. The optimality conditions are derived via the well-known Dynamic Programming Principle (DPP) introduced by R. Bellman. We first introduce the value function for an initial datum $(x, t) \in \mathbb{R}^N \times [t, T]$:

$$v(x, t) := \inf_{u \in \mathcal{U}} J_{x,t}(u) \tag{2.6}$$

which can be represented via the DPP, i.e. for every $\tau \in [t, T]$:

$$v(x, t) = \inf_{u \in \mathcal{U}} \left\{ \int_t^\tau L(y(s), u(s), s) ds + v(y(\tau), \tau) \right\}. \tag{2.7}$$

Due to (2.7) the HJB can be derived for every $x \in \mathbb{R}^N$, $s \in [t, T)$:

$$\begin{cases} -\dfrac{\partial v}{\partial s}(x, s) + \max\limits_{u \in U} \{-L(x, u, s) - \nabla v(x, s) \cdot f(x, u, s)\} = 0, \\ v(x, T) = g(x). \end{cases} \tag{2.8}$$

Once the value function is known, by e.g. solving (2.8), then the optimal feedback control can be obtained as:

$$u^*(t) := \arg\max_{u \in U} \left\{-L(x,u,t) - \nabla v(x,t) \cdot f(x,u,t)\right\}. \tag{2.9}$$

Remark 2.1 The conditions outlined in this section are employed in [47] to establish the first-order convergence of the scheme. It is important to note that these conditions are stringent and are not satisfied by the examples presented in the section dedicated to numerical tests. The relaxation of these conditions will be a focus of future research.

2.1 Dynamic Programming on a Tree Structure

We briefly sketch the essential features of the dynamic programming approach on a tree based on the discrete approximation of the dynamical system. More details on the tree structure algorithm can be found in [3] where the algorithm and several tests have been presented.

It is hard to find analytical solutions of the HJB Eq. (2.8) due to the nonlinearity and classical approximation methods, e.g. finite difference or semi-Lagrangian schemes, need a space discretization that is impossible to manage in high-dimension (see the book [21] for a comprehensive analysis of approximation schemes for Hamilton-Jacobi equations). This has motivated different approaches to mitigate the "curse of dimensionality".

We consider the discretized problem with a time step $\Delta t := [(T-t)/N_t]$ where N_t is the number of temporal time steps

$$\begin{cases} V^n(x) = \min\limits_{u \in U}[\Delta t\, L(x,u,t_n) + V^{n+1}(x + \Delta t f(x,u,t_n))], \quad n = N_t - 1, \dots, 0, \\ V^{N_t}(x) = g(x), \quad x \in \mathbb{R}^N, \end{cases} \tag{2.10}$$

where $t_n = t + n\Delta t,\ t_{N_t} = T$, and $V^n(x) := V(x,t_n)$. The classical approach computes the solution through the application of an interpolation operator to obtain the term $V^{n+1}(x + \Delta t f(x,u,t_n))$ based on the values sitting on the grid nodes. This direction will be abandoned to build a tree structure and computing (2.10) only on a tree structure. Starting from the initial condition x, we consider all the nodes obtained following the discrete dynamics, e.g. for the explicit Euler scheme with different discrete controls u_j. This gives in one step the points

$$\zeta_j^1 = x + \Delta t\, f(x, u_j, t_0), \qquad j = 1, \dots, M. \tag{2.11}$$

We assume that the control set U is a hypercube in $\mathbb{R}^m$ discretized in all directions with constant step-size Δu, obtaining a discrete control set with a finite number of points $U^{\Delta u} = \{u_1, \dots, u_M\}$ that in the sequel we continue to denote by U (with a slight abuse of notation).

Therefore, from every point x we can reach M points by (2.11). Identifying the root of the tree with $\mathcal{T}^0 = \{x\}$ we obtain the first level of the tree $\mathcal{T}^1 = \{\zeta_1^1, \dots, \zeta_M^1\}$. We can proceed in this way so that all the nodes at the $n-$th *time level*, will be given by

$$\mathcal{T}^n = \{\zeta_i^{n-1} + \Delta t f(\zeta_i^{n-1}, u_j, t_{n-1});\ j = 1, \dots, M,\ i = 1, \dots, M^{n-1}\}$$

and all the nodes belonging to the tree can be shortly defined as

$$\mathcal{T} := \{\zeta_j^n;\ j = 1, \dots, M^n,\ n = 0, \dots, N_t\},$$

where the nodes ζ_i^n are the result of the dynamics at time t_n with the controls $\{u_{j_k}\}_{k=0}^{n-1}$:

$$\zeta_{i_n}^n = \zeta_{i_{n-1}}^{n-1} + \Delta t f(\zeta_{i_{n-1}}^{n-1}, u_{j_{n-1}}, t_{n-1}) = x + \Delta t \sum_{k=0}^{n-1} f(\zeta_{i_k}^k, u_{j_k}, t_k),$$

with $\zeta^0 = x$, $i_k = \left\lfloor \frac{i_{k+1}}{M} \right\rfloor$ and $j_k \equiv i_{k+1} \mathrm{mod}\, M$, where $\zeta_i^k \in \mathbb{R}^N$, $i = 1, \ldots, M^k$ and $\lfloor \cdot \rfloor$ is the ceiling function.

Despite the fact that the tree structure allows the resolution of high dimensional problems, the construction may be expensive since $|\mathcal{T}| = O(M^{N_t})$, where N_t the number of time steps and M is the number of controls. Whenever M or N_t are too large, the construction turns out to be infeasible due to the memory allocation. In Sect. 5 we will introduce two pruning criteria and theoretical results on the reduction of the cardinality, showing their efficiency in avoiding the allocation memory problem.

Once the tree structure $\mathcal{T}$ has been constructed, we compute the numerical value function $V(x, t)$ on the tree nodes as

$$V(x, t_n) = V^n(x), \quad \forall x \in \mathcal{T}^n, \tag{2.12}$$

where $t_n = t + n\Delta t$. It is now straightforward to evaluate the value function. Since the TSA defines a grid $\mathcal{T}^n = \{\zeta_j^n\}_{j=1}^{M^n}$ for $n = 0, \ldots, N_t$, we can approximate (2.8) as follows:

$$\begin{cases} V^n(\zeta_i^n) = \min\limits_{u \in U}\{V^{n+1}(\zeta_i^n + \Delta t f(\zeta_i^n, u, t_n)) + \Delta t\, L(\zeta_i^n, u, t_n)\}, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \zeta_i^n \in \mathcal{T}^n, n = N_t - 1, \ldots, 0, \\ V^{N_t}(\zeta_i^{N_t}) = g(\zeta_i^{N_t}), \qquad\qquad\qquad\qquad\quad \zeta_i^{N_t} \in \mathcal{T}^{N_t}, \end{cases} \tag{2.13}$$

where the minimization is computed by comparison on the discretized set of controls U.

Remark 2.2 In the current framework, the curse of dimensionality in the state space has been exchanged for the curse of dimensionality in the control space and the number of time steps. The aforementioned pruning criteria help to alleviate these limitations; however, as the dimension of the control set increases, the complexity of the TSA becomes excessively demanding. Our primary focus is on low-dimensional controls, a framework of significant interest in practical applications, as it involves minimizing a specific cost functional with the addition of only a few inputs into the dynamics.

3 Reduced Order Models on a Tree Structure

Despite the fact that the tree structure algorithm avoids the construction of a grid in high dimensions, the resulting memory requirements can still be overwhelming. A first step towards relieving this computational demand via model order reduction was presented in [7]. More precisely, the POD-DEIM algorithm from [13, 20] is used to reduce the dimension of the discrete dynamical system, so that the the tree construction is performed in low dimension. Nevertheless, the POD-DEIM algorithm itself has some computational drawbacks. Firstly, if the discrete dynamical system from the finite difference semi-discretization of a PDE in dimension d, the memory requirements in both the offline and online phases of POD-DEIM are of $\mathcal{O}(N)$, where $N = \prod_{i=1}^d n_i$, where n_i is the number of discretization nodes in the ith spatial direction. A similar increase in memory requirements is experienced for other discretization techniques.

Instead, for high-dimensional semi-discrete PDEs, we couple the tree structure algorithm with the multilinear POD-DEIM algorithm presented in [30] for the 2D case and in [29] for higher dimensions. This will decrease the memory requirements to $\mathcal{O}(\widetilde{N})$, with $\widetilde{N} = \sum_{i=1}^d n_i$. A detailed comparative analysis of the computation complexity of POD-DEIM and multilinear POD-DEIM in the 2D setting can be found in Appendix A of [30].

To this end, we will first discuss some basic tensor notation required for the new algorithm, after which we will review the standard HJB-POD algorithm, before introducing the new multilinear one.

3.1 Notation and Tensor Basics

The third mode of a third-order tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, is given by (see e.g., [32])

$$\boldsymbol{\mathcal{T}}_{(3)} = \left(\boldsymbol{T}_1, \boldsymbol{T}_2, \cdots, \boldsymbol{T}_{n_2}\right),$$

where $\boldsymbol{T}_i \in \mathbb{R}^{n_3 \times n_1}$, $i = 1, 2, \ldots, n_2$ is referred to as a lateral slice, and $\boldsymbol{\mathcal{T}}_{(3)}$ is a matrix in $\mathbb{R}^{n_3 \times n_1 n_2}$. Multiplication between a tensor and a matrix, is done via the $m-$mode product, which, for a tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and a matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n_m}$, we express as

$$\boldsymbol{\mathcal{Q}} = \boldsymbol{\mathcal{T}} \times_m \boldsymbol{M} \quad \Longleftrightarrow \quad \boldsymbol{\mathcal{Q}}_{(m)} = \boldsymbol{M}\boldsymbol{\mathcal{T}}_{(m)},$$

in the m-th mode, for $m = 1, 2, 3$. For a $d-$dimensional tensor $\boldsymbol{\mathcal{T}}$, the m mode product with a matrix $\boldsymbol{M}_m \in \mathbb{R}^{n \times n_m}$ in all d dimensions will be expressed as

$$\boldsymbol{\mathcal{T}} \bigtimes_{m=1}^{d} \boldsymbol{M}_m.$$

The Kronecker product of two matrices $\boldsymbol{M} \in \mathbb{R}^{m_1 \times m_2}$ and $\boldsymbol{N} \in \mathbb{R}^{n_1 \times n_2}$ is defined as

$$\boldsymbol{M} \otimes \boldsymbol{N} = \begin{pmatrix} M_{1,1}\boldsymbol{N} & \cdots & M_{1,m_2}\boldsymbol{N} \\ \vdots & \ddots & \vdots \\ M_{m_1,1}\boldsymbol{N} & \cdots & M_{m_2,m_2}\boldsymbol{N} \end{pmatrix} \in \mathbb{R}^{m_1 n_1 \times m_2 n_2},$$

and the vec$(\cdot)$ operator stacks the columns of a matrix one after the other to form a long vector. For a third order tensor, the vectorization is applied via the first mode unfolding. Furthermore,

$$(\boldsymbol{M} \otimes \boldsymbol{N})\text{vec}(\boldsymbol{X}) = \text{vec}(\boldsymbol{N}\boldsymbol{X}\boldsymbol{M}^\top). \tag{3.1}$$

As a result, if $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, and $\boldsymbol{X} = \boldsymbol{\mathcal{X}}_{(3)}^\top$, then

$$(\boldsymbol{L} \otimes \boldsymbol{M} \otimes \boldsymbol{N})\text{vec}\left(\boldsymbol{\mathcal{X}}\right) = \text{vec}\left((\boldsymbol{M} \otimes \boldsymbol{N})\boldsymbol{X}\boldsymbol{L}^\top\right). \tag{3.2}$$

More important properties include (see, e.g., [25]): (i) $(\boldsymbol{M} \otimes \boldsymbol{N})^\top = \boldsymbol{M}^\top \otimes \boldsymbol{N}^\top$; (ii) $(\boldsymbol{M}_1 \otimes \boldsymbol{N}_1)(\boldsymbol{M}_2 \otimes \boldsymbol{N}_2) = (\boldsymbol{M}_1\boldsymbol{M}_2 \otimes \boldsymbol{N}_1\boldsymbol{N}_2)$; and (iii) $\|\boldsymbol{M} \otimes \boldsymbol{N}\|_2 = \|\boldsymbol{M}\|_2\|\boldsymbol{N}\|_2$.

3.2 POD-DEIM Reduced Dynamics

Consider the nonlinear dynamical system (2.1). In what follows we will assume without loss of generality that the linear and nonlinear terms on the right-hand side can be explicitly separated to yield a semilinear system of the form

$$\begin{cases} \dot{\boldsymbol{y}}(s) = f(\boldsymbol{y}(s), u(s), s) := \boldsymbol{L}\boldsymbol{y}(s) + \boldsymbol{f}(\boldsymbol{y}(s), u(s), s), & s \in (t, T], \\ y(t) = x \in \mathbb{R}^N, \end{cases} \tag{3.3}$$

where $\boldsymbol{L} \in \mathbb{R}^{N \times N}$ and $\boldsymbol{f} : \mathbb{R}^N \times \mathbb{R}^m \times [t, T] \to \mathbb{R}^N$ is a continuous function in all arguments and locally Lipschitz-type with respect to the first variable.

In [7] the authors reduce the number of variables involved in the system (3.3) by means of POD-DEIM. More precisely, given n_s time instances in the timespan $(t, T]$ and M discrete controls, consider constructing a tree structure as described in Sect. 2.1 to obtain a set of solution *snapshots* $\mathcal{T} = \{\zeta_j^n;\ j = 1, \dots, M^n,\ n = 0, \dots, n_s\}$, *i.e.* all the nodes of the tree, and consider the *snapshot matrix*

$$\boldsymbol{S} = [\zeta_1^0, \zeta_1^1, \dots, \zeta_1^M, \dots, \zeta_1^{n_s}, \dots, \zeta_{M^{n_s}}^{n_s}] \in \mathbb{R}^{N\times|\mathcal{T}|}, \qquad \mathcal{S} = \text{Range}(\boldsymbol{S}).$$

A POD basis of dimension $k \leq |\mathcal{T}|$ is obtained by orthogonal reduction of the matrix $\boldsymbol{S}$. That is, given the Singular Value Decomposition (SVD)

$$\boldsymbol{S} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{W}^\top, \qquad \boldsymbol{V}, \boldsymbol{W} \in \mathbb{R}^{N\times|\mathcal{T}|}, \boldsymbol{\Sigma} \in \mathbb{R}^{n_s\times n_s},$$

the POD basis is given by $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_k\}$, where $\boldsymbol{V}_k = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_k] \in \mathbb{R}^{N\times k}$ is the matrix of truncated left singular vectors related to the k largest singular values contained on the diagonal of $\boldsymbol{\Sigma}$.

Given the matrix $\boldsymbol{V}_k$, the state vector $\boldsymbol{y}(s)$ can be approximated as $\boldsymbol{y}(s) \approx \boldsymbol{V}_k\widehat{\boldsymbol{y}}(s)$, for all $s \in (t, T]$, where $\widehat{\boldsymbol{y}}(s) \in \mathbb{R}^k$ solves the reduced dynamical system

$$\begin{cases} \dot{\widehat{y}}(s) = \boldsymbol{V}_k^\top f(\boldsymbol{V}_k\widehat{\boldsymbol{y}}(s), u(s), s) := \boldsymbol{V}_k^\top \boldsymbol{L}\boldsymbol{V}_k\widehat{\boldsymbol{y}}(s) + \boldsymbol{V}_k^\top \boldsymbol{f}(\boldsymbol{V}_k\widehat{\boldsymbol{y}}(s), u(s), s), \\ \widehat{y}(t) = \boldsymbol{V}_k^\top x. \end{cases} \tag{3.4}$$

To ensure that the reduced model can be simulated with a computational cost independent of N, we need to avoid lifting the nonlinear term before projection onto the low-dimensional space. Consequently, the Discrete Empirical Interpolation Method (DEIM) from [13] is used to interpolate the nonlinear function.

To this end we consider an approximation of the form

$$\boldsymbol{f}(\boldsymbol{V}_k\widehat{\boldsymbol{y}}(s), u(s), s) \approx \boldsymbol{\Phi}_p\widehat{\boldsymbol{f}}(\boldsymbol{V}_k\widehat{\boldsymbol{y}}(s), u(s), s), \qquad \widehat{\boldsymbol{f}}(\boldsymbol{V}_k\widehat{\boldsymbol{y}}(s), u(s), s) \in \mathbb{R}^p,$$

where $\boldsymbol{\Phi}_p = [\boldsymbol{\varphi}_1, \dots, \boldsymbol{\varphi}_p] \in \mathbb{R}^{N\times p}$, with $p \ll N$, and $\{\boldsymbol{\varphi}_1, \dots, \boldsymbol{\varphi}_p\}$ is a POD basis of dimension p obtained from evaluating $\boldsymbol{f}$ on the set of snapshots $\mathcal{T}$. The overdetermined system is solved by interpolation, ensuring that the left and right side of the equation is equal at p selected points. That is, given the matrix $\boldsymbol{P} = [\boldsymbol{e}_{\rho_1}, \dots, \boldsymbol{e}_{\rho_p}] \in \mathbb{R}^p$ containing a subset of columns of the identity matrix, we ensure that $\boldsymbol{P}^\top\boldsymbol{f}(\boldsymbol{V}_k\widehat{\boldsymbol{y}}(s), u(s), s) = \boldsymbol{P}^\top\boldsymbol{\Phi}_p\widehat{\boldsymbol{f}}(\boldsymbol{V}_k\widehat{\boldsymbol{y}}(s), u(s), s)$, so that

$$\boldsymbol{f}(\boldsymbol{V}_k\widehat{\boldsymbol{y}}(s), u(s), s) \approx \widetilde{\boldsymbol{f}}(\widehat{\boldsymbol{y}}(s), u(s), s) := \boldsymbol{\Phi}_p(\boldsymbol{P}^\top\boldsymbol{\Phi}_p)^{-1}\boldsymbol{P}^\top\boldsymbol{f}(\boldsymbol{V}_k\widehat{\boldsymbol{y}}(s), u(s), s).$$

Throughout this paper we deal with nonlinear functions that are evaluated element-wise, so that

$$\widetilde{\boldsymbol{f}}(\widehat{\boldsymbol{y}}(s), u(s), s) = \boldsymbol{\Phi}_p(\boldsymbol{P}^\top\boldsymbol{\Phi}_p)^{-1}\boldsymbol{f}(\boldsymbol{P}^\top\boldsymbol{V}_k\widehat{\boldsymbol{y}}(s), u(s), s),$$

and the nonlinear term is only evaluated at p entries.

3.3 A Multilinear HJB-POD-DEIM Algorithm on a Tree Structure

In this section we illustrate how, under certain hypotheses, the discrete system (3.3) can be expressed, integrated and reduced in terms of multilinear arrays; see e.g., [17, 29, 41, 49]. We focus specifically on the case where the discrete system (3.3) stems from the space discretization of a semilinear PDE of the form

$$\begin{cases} \partial_s y(s,x) = \mathcal{L}(y(s,x)) + \boldsymbol{f}(\nabla y(s,x), y(s,x), u(s), s), \\ \qquad s \in (t, T], x \in \Omega, \\ y(t,x) = \tilde{y}(x), \quad x \in \Omega, \end{cases} \tag{3.5}$$

with $\mathcal{L}$ a linear differential operator, $\boldsymbol{f}$ a generic nonlinear operator and $\Omega \subset \mathbb{R}^d$, for $d = 2, 3$.

3.3.1 Discretization in Terms of Multilinear Arrays

Consider the operator $\mathcal{L}$ to be a second order differential operator with separable coefficients. Then, the physical domain can be mapped to a hypercubic domain $\boldsymbol{\Omega} = [a_1, b_1] \times \cdots \times [a_d, b_d]$, if the operator is discretized via a tensor basis. Examples of such discretizations include, but are not limited to, spectral methods, and finite differences on parallelepipedal domains. See, e.g., [29, 41, 49] for more information regarding the assumptions on the operators, domains and discretization techniques. Here we consider $\mathcal{L}$ as a $d-$dimensional Laplace operator for illustration purposes, but more general operators can also be treated. Under these conditions, it holds (from (3.3)) that

$$\boldsymbol{L} = \sum_{m=1}^{d} \boldsymbol{I}_{n_d} \otimes \cdots \otimes \boldsymbol{A}_m \otimes \cdots \otimes \boldsymbol{I}_{n_1} \in \mathbb{R}^{N \times N},$$

where $\boldsymbol{A}_m \in \mathbb{R}^{n_m \times n_m}$ contains the approximation of the second derivative in the x_m direction. We will also consider problems where the nonlinear term $\boldsymbol{f}$ depends on the first derivative of the state vector, so that we also define the matrix

$$\boldsymbol{D} = \sum_{m=1}^{d} \boldsymbol{I}_{n_d} \otimes \cdots \otimes \boldsymbol{B}_m \otimes \cdots \otimes \boldsymbol{I}_{n_1} \in \mathbb{R}^{N \times N},$$

where $\boldsymbol{B}_m \in \mathbb{R}^{n_m \times n_m}$ contains the approximation of the first derivative in the x_m direction. The vectors $\boldsymbol{y}(t) \in \mathbb{R}^N$ from (3.3) then represent the vectorization of the elements of a tensor $\boldsymbol{\mathcal{Y}}(t) \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, such that $\boldsymbol{y}(t) = \mathtt{vec}(\boldsymbol{\mathcal{Y}}(t))$, $\boldsymbol{L}\boldsymbol{y} = \mathrm{vec}\,(\mathcal{A}(\boldsymbol{\mathcal{Y}}))$ and $\boldsymbol{D}\boldsymbol{y} = \mathrm{vec}\,(\mathcal{D}(\boldsymbol{\mathcal{Y}}))$, where[1]

$$\mathcal{A}(\boldsymbol{\mathcal{Y}}) := \sum_{m=1}^{d} \boldsymbol{\mathcal{Y}} \times_m \boldsymbol{A}_m \quad \text{and} \quad \mathcal{D}(\boldsymbol{\mathcal{Y}}) := \sum_{m=1}^{d} \boldsymbol{\mathcal{Y}} \times_m \boldsymbol{B}_m. \tag{3.6}$$

We refer the reader back to Sect. 3.1 for a description of how the $\mathtt{vec}$ operator relates the entries of the tensor $\boldsymbol{\mathcal{Y}}(t)$ and the vector $\boldsymbol{y}(t)$. Moreover, if the function $\mathcal{F} : \mathcal{S} \times [0, t_f] \to \mathbb{R}^{n_1 \times \cdots \times n_d}$ represents the function $\boldsymbol{f}$ evaluated at the entries of the array $\boldsymbol{\mathcal{Y}}$ and $\mathcal{D}(\boldsymbol{\mathcal{Y}})$, then it holds that $\boldsymbol{f}(\boldsymbol{D}\boldsymbol{y}, \boldsymbol{y}(s), u(s), s) = \mathtt{vec}\,(\mathcal{F}\,(\mathcal{D}(\boldsymbol{\mathcal{Y}}), \boldsymbol{\mathcal{Y}}, u(s), s))$, and (3.3) can be written in the form

$$\begin{cases} \dot{\boldsymbol{\mathcal{Y}}}(s) & = \mathcal{A}(\boldsymbol{\mathcal{Y}}(s)) + \mathcal{F}\,(\mathcal{D}(\boldsymbol{\mathcal{Y}}(s)), \boldsymbol{\mathcal{Y}}(s), u(s), s)\,, \\ \boldsymbol{\mathcal{Y}}(t) & = \boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}. \end{cases} \tag{3.7}$$

The boundary conditions are contained in the matrices $\boldsymbol{A}_m$ and $\boldsymbol{B}_m$, $m = 1, \ldots, d$; see e.g., [17, 41]. From here on forward we consider the case where $n_1 = \cdots = n_d = n$, so that $N = n^d$.

3.3.2 Higher-Order POD (HO-POD) Model Reduction

As it has been shown in [29], great savings in terms of memory requirements and computational time can be obtained by applying model order reduction directly to the system (3.7)

[1] For the case $d = 2$, (3.6) are Sylvester operators of the form $\boldsymbol{A}_1\boldsymbol{Y} + \boldsymbol{Y}\boldsymbol{A}_2^\top$ and $\boldsymbol{B}_1\boldsymbol{Y} + \boldsymbol{Y}\boldsymbol{B}_2^\top$ respectively [49]. If $d = 3$, these operators can respectively be expressed as $\boldsymbol{\mathcal{Y}} \times_1 \boldsymbol{A}_1 + \boldsymbol{\mathcal{Y}} \times_2 \boldsymbol{A}_2 + \boldsymbol{\mathcal{Y}} \times_3 \boldsymbol{A}_3$ and $\boldsymbol{\mathcal{Y}} \times_1 \boldsymbol{B}_1 + \boldsymbol{\mathcal{Y}} \times_2 \boldsymbol{B}_2 + \boldsymbol{\mathcal{Y}} \times_3 \boldsymbol{B}_3$.

instead of first vectorizing and applying model reduction to the vectorized system (3.3). To this end, we consider an approximation of the tensor $\boldsymbol{\mathcal{Y}}(s)$ of the form

$$\boldsymbol{\mathcal{Y}}(s) \approx \widetilde{\boldsymbol{\mathcal{Y}}}(s) := \widehat{\boldsymbol{\mathcal{Y}}}(s) \bigtimes_{m=1}^{d} \boldsymbol{V}_m,$$

where $\boldsymbol{V}_m \in \mathbb{R}^{n_m \times k_m}$ are tall matrices with orthonormal columns and $\widehat{\boldsymbol{\mathcal{Y}}}(s) \in \mathbb{R}^{k_1 \times \cdots \times k_d}$ ($k_m \ll n$, with $m = 1, 2, 3 \ldots, d$) satisfies the low-dimensional equation

$$\begin{cases} \dot{\widehat{\boldsymbol{\mathcal{Y}}}}(s) & = \widehat{\mathcal{A}}(\widehat{\boldsymbol{\mathcal{Y}}}(s)) + \widehat{\mathcal{F}}\left(\widehat{\mathcal{D}}(\widehat{\boldsymbol{\mathcal{Y}}}(s)), \widehat{\boldsymbol{\mathcal{Y}}}(s), u(s), s\right), \\ \widehat{\boldsymbol{\mathcal{Y}}}(t) & = \boldsymbol{\mathcal{X}} \times_{m=1}^{d} \boldsymbol{V}_m^\top \in \mathbb{R}^{k_1 \times \cdots \times k_d}, \end{cases} \tag{3.8}$$

where

$$\widehat{\mathcal{F}}\left(\widehat{\mathcal{D}}(\widehat{\boldsymbol{\mathcal{Y}}}(s)), \widehat{\boldsymbol{\mathcal{Y}}}(s), u(s), s\right) = \mathcal{F}\left(\mathcal{D}(\widetilde{\boldsymbol{\mathcal{Y}}}(s)), \widetilde{\boldsymbol{\mathcal{Y}}}(s), u(s), s\right) \bigtimes_{m=1}^{d} \boldsymbol{V}_m^\top \tag{3.9}$$

and

$$\widehat{\mathcal{A}}(\widehat{\boldsymbol{\mathcal{Y}}}) := \sum_{m=1}^{d} \widehat{\boldsymbol{\mathcal{Y}}} \times_m \widehat{\boldsymbol{A}}_m, \quad \widehat{\boldsymbol{A}}_m = \boldsymbol{V}_m^\top \boldsymbol{A}_m \boldsymbol{V}_m. \tag{3.10}$$

The matrices $\boldsymbol{V}_m \in \mathbb{R}^{n_m \times k_m}$ can be obtained via the HO-POD algorithm described in [29]. That is, given a set of snapshots $\{\boldsymbol{\mathcal{Y}}(s_i)\}_{i=1}^{n_s}$, each matrix $\mathbf{V}_m$ is constructed in order to approximate the left range space of the matrix

$$\boldsymbol{\mathcal{S}}_{(m)} = \left(\boldsymbol{\mathcal{Y}}_{(m)}(s_1), \ldots, \boldsymbol{\mathcal{Y}}_{(m)}(s_{n_s})\right) \in \mathbb{R}^{n \times n^{d-1} n_s}, \quad \text{for} \quad m = 1, \ldots, d,$$

where m represents the mode along which the tensor is unfolded to form the matrices $\boldsymbol{\mathcal{Y}}_{(m)}(s_i)$, $i = 1, \ldots, n_s$, and $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{n \times \cdots \times n}$ is a tensor of order d containing the snapshots. Note that neither the matrices $\boldsymbol{\mathcal{S}}_{(m)}$ or the tensor $\boldsymbol{\mathcal{S}}$ is ever explicitly constructed or stored. Instead we follow the dynamic algorithm initially introduced in [30] for approximating the left range space of $\boldsymbol{\mathcal{S}}_{(m)}$. In [29] a simpler algorithm was used to construct the approximation space in the tensor setting. Here we implement the more refined dynamic algorithm for the tensor setting; the inclusion of snapshot information is discussed here, whereas snapshot selection will be presented in Sect. 4.

Suppose κ[2] is the maximum admissible dimension for the reduced space in all modes, selected a-priori, and consider the initial condition $\boldsymbol{\mathcal{Y}}(t)$. Let

$$\boldsymbol{\mathcal{Y}}(t) \approx \boldsymbol{\mathcal{C}}(t) \bigtimes_{m=1}^{d} \boldsymbol{U}_m^{(0)}, \tag{3.11}$$

represent the sequentially truncated higher order SVD[3] (STHOSVD) [54] of $\boldsymbol{\mathcal{Y}}(t)$, where $\boldsymbol{U}_m^{(0)}$ contains the first κ dominant left singular vectors of $\boldsymbol{\mathcal{Y}}_{(m)}(t)$. For each mode these left singular vectors are collected into the matrix $\widetilde{\boldsymbol{V}}_m = \boldsymbol{U}_m^{(0)}$, $m = 1, \ldots, d$.

Subsequently, suppose the snapshot at time instance s_j has been selected for inclusion into the approximation space and let

$$\boldsymbol{\mathcal{Y}}(s_j) \approx \boldsymbol{\mathcal{C}}(s_j) \bigtimes_{m=1}^{d} \boldsymbol{U}_m^{(j)}, \tag{3.12}$$

[2] We refer the reader to [30] for a detailed experimental analysis on the role of the parameter κ.

[3] For the case $d = 2$, however, we just use the standard MATLAB SVD function.

represent the STHOSVD of the selected snapshot and let $\boldsymbol{\Sigma}_m^{(j)}$ contain the first κ singular values of $\boldsymbol{\mathcal{Y}}_{(m)}(s_j)$ on the main diagonal. The approximation spaces are updated by appending the new singular values and vectors, so that

$$\widetilde{\boldsymbol{V}}_m \leftarrow [\widetilde{\boldsymbol{V}}_m, \boldsymbol{U}_m^{(j)}], \quad \text{and} \quad \widetilde{\boldsymbol{\Sigma}}_m \leftarrow \texttt{blkdiag}(\widetilde{\boldsymbol{\Sigma}}_m, \boldsymbol{\Sigma}_m^{(j)}).$$

Eventually the diagonal entries of $\widetilde{\boldsymbol{\Sigma}}_m$ are reordered decreasingly and truncated so that the largest κ values are retained, with the vectors in $\widetilde{\boldsymbol{V}}_m$ reordered and truncated accordingly. This procedure is summarized in Algorithm 1.

At the end of the procedure, when all snapshots have been processed, the final basis vectors are obtained by orthogonal reduction of the matrices $\widetilde{\boldsymbol{V}}_m$. More precisely, let $\widetilde{\boldsymbol{V}}_m = \overline{\boldsymbol{V}}_m \,\overline{\boldsymbol{\Sigma}}_m \,\overline{\boldsymbol{W}}_m^\top$ be the SVD of $\widetilde{\boldsymbol{V}}_m$. The final basis matrices $\boldsymbol{V}_m$ are obtained by truncating the first k_m dominant singular vectors of $\overline{\boldsymbol{V}}_m$ according to the criterion

$$\sqrt{\sum_{i=k_m+1}^{\kappa} (\sigma_m^{(i)})^2} < \tau \sqrt{\sum_{i=1}^{\kappa} (\sigma_m^{(i)})^2}, \tag{3.13}$$

for some $\tau \in (0, 1)$, where $\sigma_m^{(i)}$ is the i-th diagonal element of $\overline{\boldsymbol{\Sigma}}_m$.

Algorithm 1 Appending a new snapshot to the constructed space

```
1: Given $\widetilde{\boldsymbol{V}}_m \in \mathbb{R}^{n\times\kappa}$, $\widetilde{\boldsymbol{\Sigma}}_m \in \mathbb{R}^{\kappa\times\kappa}$, and a new snapshot $\boldsymbol{\mathcal{Y}}(s_j) \in \mathbb{R}^{n\times\cdots\times n}$ for $m = 1, \ldots, d$
2: Compute the STHOSVD of $\boldsymbol{\mathcal{Y}}(s_j)$ to obtain (3.12)
3: for $m = 1, \ldots, d$ do
4:     Let $\boldsymbol{\Sigma}_m^{(j)}$ contain the first $\kappa$ singular values of $\boldsymbol{\mathcal{Y}}_{(m)}(s_j)$ on the main diagonal
5:     Append: $\widetilde{\boldsymbol{V}}_m \leftarrow [\widetilde{\boldsymbol{V}}_m, \boldsymbol{U}_m^{(j)}]$ & $\widetilde{\boldsymbol{\Sigma}}_m \leftarrow \texttt{blkdiag}(\widetilde{\boldsymbol{\Sigma}}_m, \boldsymbol{\Sigma}_m^{(j)})$.
6:     Decreasingly order the entries of $\widetilde{\boldsymbol{\Sigma}}_m$ and keep the first $\kappa$
7:     Order $\widetilde{\boldsymbol{V}}_m$ accordingly and keep the first $\kappa$ vectors of each;
8:     Output the new space vectors $\widetilde{\boldsymbol{V}}_m \in \mathbb{R}^{n\times\kappa}$ and singular values $\widetilde{\boldsymbol{\Sigma}}_m \in \mathbb{R}^{\kappa\times\kappa}$
9: end for
```

3.3.3 Higher-Order DEIM (HO-DEIM) Approximation of the Nonlinear Term

It is clear from (3.9) that a bottleneck forms around the reduced nonlinear term, similar to the vector setting. To this end, we consider the HO-DEIM algorithm from [29] to circumvent the issue. Consequently, suppose the tall matrices $\boldsymbol{\Phi}_m \in \mathbb{R}^{n_m\times p_m}$, for $m = 1, 2, \ldots, d$, have been constructed as the output of the HO-POD method described above for the snapshots $\{\mathcal{F}(\mathcal{D}(\boldsymbol{\mathcal{Y}}(s_i)), \boldsymbol{\mathcal{Y}}(s_i), u(s_i), s_i)\}_{i=1}^{n_s}$. Furthermore, consider d matrices $\boldsymbol{P}_m \in \mathbb{R}^{n_m\times p_m}$ each containing a subset of columns of the $n_m \times n_m$ identity matrix. The matrices $\boldsymbol{P}_m$ are each respectively obtained as the output of the `q-deim` algorithm [20] with input $\boldsymbol{\Phi}_m^\top$. The HO-DEIM approximation of (3.9) is then given by

$$\begin{aligned}\widehat{\mathcal{F}}\left(\widehat{\mathcal{D}}(\widehat{\boldsymbol{\mathcal{Y}}}(s)), \widehat{\boldsymbol{\mathcal{Y}}}(s), u(s), s\right) &\approx \mathcal{F}\left(\mathcal{D}(\widetilde{\boldsymbol{\mathcal{Y}}}(s)), \widetilde{\boldsymbol{\mathcal{Y}}}(s), u(s), s\right) \bigtimes_{m=1}^{d} \mathbf{F}_m \\ &= \widehat{\mathcal{F}}^{\text{DEIM}}\left(\widehat{\mathcal{D}}(\widehat{\boldsymbol{\mathcal{Y}}}(s)), \widehat{\boldsymbol{\mathcal{Y}}}(s), u(s), s\right),\end{aligned} \tag{3.14}$$

where

$$\mathbf{F}_m = \mathbf{V}_m^\top \Phi_m (\mathbf{P}_m^\top \Phi_m)^{-1} \mathbf{P}_m^\top.$$

If $\mathcal{F}$ is evaluated element-wise at the components of $\widetilde{\boldsymbol{\mathcal{Y}}}(s)$ and $\mathcal{D}(\widetilde{\boldsymbol{\mathcal{Y}}}(s))$, then it holds that

$$\begin{aligned}\widetilde{\mathcal{F}\left(\mathcal{D}(\widetilde{\boldsymbol{\mathcal{Y}}}(s)), \widetilde{\boldsymbol{\mathcal{Y}}}(s), u(s), s\right)} &:= \mathcal{F}\left(\mathcal{D}(\widetilde{\boldsymbol{\mathcal{Y}}}(s)), \widetilde{\boldsymbol{\mathcal{Y}}}(s), u(s), s\right) \bigtimes_{m=1}^{d} \mathbf{P}_m^\top \\ &= \mathcal{F}\left(\mathcal{D}(\widetilde{\boldsymbol{\mathcal{Y}}}(s)) \bigtimes_{m=1}^{d} \mathbf{P}_m^\top,\ \widetilde{\boldsymbol{\mathcal{Y}}}(s) \bigtimes_{m=1}^{d} \mathbf{P}_m^\top,\ u(s), s\right).\end{aligned} \tag{3.15}$$

In this case the nonlinear term is evaluated at only $p_1 p_2 \cdots p_m$ entries.

3.3.4 The Reduced Optimal Control Problem on a Tree Structure

In this section we explore how the full procedure combining the tree structure algorithm and the HO-POD-DEIM Model reduction technique is split into an *offline* and *online* stage to solve the HJB Eq. (2.8) and determine the optimal control (2.9).

Offline Stage The offline stage consists of two important steps, namely *snapshot collection* and *basis construction*. To this end, we select a coarse time step $\widehat{\Delta t}$ and control set $\widehat{U}$. The basis is constructed on the fly following Sects. 3.3.2–3.3.3, on the nodes of the tree, which is constructed following Sect. 2.1. This is a computationally expensive step, as the full-dimensional space is explored in this phase. To this end, we discuss a collection of nuances related to the implementation in Sect. 4.

Online Stage At this stage, the computed basis vectors are exploited to construct a reduced dimensional tree, approximate the reduced value function and compute the optimal trajectory at a fraction of the initial cost.

- **Construction of the reduced tree.** Here we fix the desired wider discrete control set $\widehat{U} \subset \widetilde{U}$ and/or a smaller time step $\Delta t \leq \widehat{\Delta t}$ for the resolution of the HO-POD-DEIM reduced dynamical system

$$\begin{cases} \dot{\widehat{\boldsymbol{\mathcal{Y}}}}(s) &= \widehat{\mathcal{A}}(\widehat{\boldsymbol{\mathcal{Y}}}(s)) + \widehat{\mathcal{F}}^{\text{DEIM}}\left(\widehat{\mathcal{D}}(\widehat{\boldsymbol{\mathcal{Y}}}(s)), \widehat{\boldsymbol{\mathcal{Y}}}(s), u(s), s\right), \\ \widehat{\boldsymbol{\mathcal{Y}}}(t) &= \boldsymbol{\mathcal{X}} \times_{m=1}^{d} \mathbf{V}_m^\top \in \mathbb{R}^{k_1 \times \cdots \times k_d}. \end{cases} \tag{3.16}$$

  Following Sect. 2.1, we build the reduced tree $\widehat{\mathcal{T}}$ as done for the offline stage. The cardinality of tree, however, still grows exponentially, despite the reduced dimension of the dynamical system. As a result we analyze a collection of important pruning criteria in Sect. 5.1 in an attempt to reduce the cardinality of the tree dynamically during the construction.
- **Approximation of the reduced value function.** The value function computed in the reduced space will be denoted by $\widehat{V}(\widehat{x}, t)$ and its approximation at time t_n as $\widehat{V}^n(\widehat{x})$. Its resolution will follow the classical scheme introduced in Sect. 2.1:

$$\begin{cases} \widehat{V}^n(\widehat{\zeta}_i^n) = \min\limits_{u \in \widetilde{U}} \{\widehat{V}^{n+1}(T^{n+1}(\widehat{\zeta}_i^n, u) + \Delta t\, \widehat{L}(\widehat{\zeta}_i^n, u, t_n)\}, \\ \qquad\qquad\qquad\qquad \widehat{\zeta}_i^n \in \widehat{\mathcal{T}}^n, n = N_t - 1, \ldots, 0, \\ \widehat{V}^{N_t}(\widehat{\zeta}_i^{N_t}) = \widehat{g}(\widehat{\zeta}_i^{N_t}), \qquad\qquad \widehat{\zeta}_i^{N_t} \in \widehat{\mathcal{T}}^{N_t}, \end{cases} \tag{3.17}$$

  where

$$\widehat{L}(\widehat{\zeta}, u, t) = L\left(\widehat{\zeta} \bigtimes_{m=1}^{d} \mathbf{V}_m^\top, u, t\right),$$

$$\widehat{g}(\widehat{\zeta}) = g\left(\widehat{\zeta}_i^{N_t} \bigtimes_{m=1}^{d} \boldsymbol{V}_m^\top\right)$$

and $T^{n+1}(\widehat{\zeta}, u)$ stands for the time evolution of the node $\widehat{\zeta}$ with control u at time t_{n+1}.

- **Computation of the optimal trajectory.**
  The optimal trajectory can be seen as a specific path in the tree structure. For this reason during the computation of the value function we store the minimizing indices in (3.17). Once completed the computation of the value function, the optimal path will be given just following the tree branches which returns the minimum index.

Remark 3.1 It is important to note that, generally, a control in feedback form cannot be derived directly at the nodes of the tree without requiring interpolation operators. For example, interpolation methods for scattered data can be employed to reconstruct the value function at nodes that are not part of the tree. A preliminary work in this direction can be found in [8].

4 Hints for the Implementation

In both the offline and online phases of the procedure, great savings in terms of CPU time and memory requirements can be obtained if implemented in an efficient way. Consequently, in this section we discuss how the snapshots are selected and how the simulated tree nodes can be efficiently stored to save on memory requirements in the offline phase. Moreover, we discuss how the reduced model can be efficiently simulated at many time steps and control inputs, to avoid high computational costs in the online phase.

4.1 Snapshot Selection

The full order model is simulated on a coarse timegrid with two control inputs which are the two extremes of the control domain, as discussed in [7]. To avoid excessive computational work we only include information from snapshots that are not yet well approximated in the current basis. The condition for snapshot inclusion is given by the projection error, that is:

$$\text{if:} \quad \frac{\|\boldsymbol{\mathcal{Y}}(s_i) - \boldsymbol{\mathcal{Y}}(s_i) \times_{m=1}^{d} \widetilde{\boldsymbol{V}}_m \widetilde{\boldsymbol{V}}_m^\top \|_F}{\|\boldsymbol{\mathcal{Y}}(s_i)\|_F} > \tau \quad \text{then:} \quad \textbf{Include}, \tag{4.1}$$

for some $\tau \in (0, 1)$. Here the matrices $\widetilde{\boldsymbol{V}}_m$ contain the current basis vectors in all modes, updated dynamically with snapshot information from the previous selected snapshots as described in Sect. 3.3.2. The full procedure is summarized in Algorithm 2 below.

Algorithm 2 Snapshot collection and basis adaptation

```
1: Given an initial snapshot $\boldsymbol{\mathcal{Y}}(t) \in \mathbb{R}^{n\times\cdots\times n}$
2: Compute the STHOSVD of $\boldsymbol{\mathcal{Y}}(t)$ to obtain (3.11) and the initial basis vectors $\widetilde{\boldsymbol{V}}_m \in \mathbb{R}^{n\times\kappa}$ and singular
   values $\widetilde{\boldsymbol{\Sigma}}_m \in \mathbb{R}^{\kappa\times\kappa}$ for $m = 1, \ldots, d$
3: for $i = 1, \ldots, n_s$ do
4:     Solve the full order model to obtain $\boldsymbol{\mathcal{Y}}(s_i)$
5:     Check the condition (4.1)
6:     if Include then
7:         Update $\widetilde{\boldsymbol{V}}_m$ and $\widetilde{\boldsymbol{\Sigma}}_m$ with the snapshot $\boldsymbol{\mathcal{Y}}(s_i)$ using Algorithm 1
8:     end if
9: end for
10: for $m = 1, \ldots, d$ do
11:     Compute the truncated SVD of $\widetilde{\boldsymbol{V}}_m$ using (3.13) to obtain $\boldsymbol{V}_m \in \mathbb{R}^{n\times k_m}$
12: end for
```

4.2 Efficient Memory Allocation by Low-Rank Storage of Tree Nodes

One challenge of the method presented in [7] is in terms of memory in the offline phase, since the high fidelity solutions need to be calculated and stored for several time steps and control inputs in order to form a reduced order model. In particular, the nodes of the tree $\mathcal{T}$ are vectorized and stored in a matrix $\boldsymbol{T} \in \mathbb{R}^{N\times|\mathcal{T}|}$, where $|\mathcal{T}| = O(M^{N_t})$, with M fixed as the number of control inputs and N_t as the number of time steps. The exponential growth of the second dimension greatly limits the number of snapshots that can be stored.

In this paper, we suggest the following improvement. Since the full order model is simulated in array form, the snapshots at the resulting tree nodes are either matrices or tensors. Consequently, we can take advantage of the (possible) low-rank structure of each node. That is, we compute the STHOSVD of each computed nodal value, truncated to the first κ singular vectors in each mode, so that $\boldsymbol{\mathcal{Y}}(s_j) \approx \boldsymbol{\mathcal{C}}(s_j) \times_{m=1}^{d} \boldsymbol{U}_m^{(j)}$, with κ selected a-priori as discussed in Sect. 3.3.2. As a result, the node can be stored in low-rank form to be recalled for later computations. More precisely, we collect and store only the dominant singular vectors in each mode and the low-dimensional core tensors such that

$$\overline{\boldsymbol{U}}_m \leftarrow [\overline{\boldsymbol{U}}_m, \boldsymbol{U}_m^{(j)}] \quad \text{for} \quad m = 1, \ldots, d \quad \text{and} \quad \bar{\boldsymbol{c}} \leftarrow [\bar{\boldsymbol{c}}, \mathrm{vec}(\boldsymbol{\mathcal{C}}(s_j))].$$

When required at the next time level, the snapshots can easily be computed from its Tucker decomposition [32, Section 4]. This process allows us to store vectors of length $n_1, \ldots, n_d$ instead of $N = n_1 n_2 \cdots n_d$. The number of vectors stored depends on the rank of the considered snapshots. A further advantage is that when a snapshot is selected for inclusion into the approximation space, a HOSVD is required as discussed in Sect. 3.3.2, which will be readily available thanks to this procedure.

Furthermore, it has been observed that only the nodes from the previous level of the tree need to be stored, since the snapshots from the earlier levels are automatically processed and discarded during the HO-POD basis construction. Finally, we observe that the computation of the value function does not require the knowledge of the nodes, but only of the corresponding cost evaluation. In this way we are going to store only the corresponding scalar cost and the nodes will be erased after the computation of its tree sons.

4.3 Efficient Simulation of the Reduced Model (3.16)

An important ingredient in the success of the HO-POD-DEIM reduction procedure is the ability to integrate the reduced model (3.16) in array form without vectorization. Semi-discrete diffusion problems are typically classified as stiff, rendering explicit methods unsuitable. Conversely, when nonlinear reaction terms are present, fully implicit schemes require a nonlinear solver, such as Newton's method, at each time step. Semi-implicit approaches offer an effective compromise (see, e.g., [28]). In this paper we consider the semi-implicit Euler scheme, given that the considered model is typically associated with a stiff linear term and a nonstiff nonlinear term, but several alternatives can be considered [17, 29]. More precisely, suppose $\widehat{\boldsymbol{\mathcal{Y}}}^{(j)}$ is an approximation of $\widehat{\boldsymbol{\mathcal{Y}}}(s_j)$, then the linear system

$$(\widehat{\mathcal{I}} - \Delta t\widehat{\mathcal{A}})\widehat{\boldsymbol{\mathcal{Y}}}^{(j)} = \widehat{\boldsymbol{\mathcal{Y}}}^{(j-1)} + \Delta t\widehat{\mathcal{F}}^{\text{DEIM}}\left(\widehat{\mathcal{D}}(\widehat{\boldsymbol{\mathcal{Y}}}^{(j-1)}), \widehat{\boldsymbol{\mathcal{Y}}}^{(j-1)}, u(s_{j-1}), s_{j-1}\right) \tag{4.2}$$

needs to evaluated at each time level s_j. Once again, vectorizing the linear system (4.2) at each time step will reduce the computational gains related to the reduction in Array form. Instead, (4.2) can be solved in array form using the direct method presented in [29, 50].

5 Pruning Techniques

Although theoretically the tree structure enables to compute the solution for arbitrary high dimensional problems, since we are not restricted to the direct discretization of a domain, its construction turns to be computationally expensive, due to the exponential growth of its cardinality, For this reason in this section we are going to introduce and analyse different *pruning criteria* able to reduce the growth of the tree, but keeping the same accuracy.

5.1 Geometric Pruning

A pruning criterion based on a comparison of the nodes in euclidean norm has been introduced in [48]. More precisely, two given nodes ζ_i^n and ζ_j^n will be merged if

$$\|\zeta_i^n - \zeta_j^n\| \leq \epsilon, \quad \text{with } i \neq j \text{ and } n = 0, \ldots, N_t, \tag{5.1}$$

for a given threshold $\epsilon > 0$. To ensure first order convergence, the threshold ϵ must scale as Δt^2 (we refer to [48] for more details about the error estimates). This pruning criterion has been successfully applied to low and high dimensional problems, but the main drawback is the expensive computation of distances in high dimension. One possible solution relies on the projection of the data onto a lower dimension minimizing the variance of the data. This procedure is already encoded in the algorithm described in Sect. 3.3.4, since we are reducing the dimension of the problem keeping the main features.

5.2 Statistical Pruning

In this section we introduce a new iterative pruning criterion based on statistical information about the value function. We suppose we are starting with a certain control set U_1. First, we construct the tree $\mathcal{T}_1$ based on the control set U_1 and the value function computed on the tree will be denoted by $V_1(x, t)$. Afterwards, we refine the constructed tree based on the information on the value function: fixing a ratio $\rho \in (0, 1]$ of the nodes, we retain just those with the lowest value function, obtaining a new tree $\widetilde{\mathcal{T}}_1$. More precisely, we have that $|\widetilde{\mathcal{T}}_1| = \rho|\mathcal{T}_1|$ and for every time level $t_n \in [t, T]$ and every node $\zeta \in \mathcal{T}_1$ there exists a node $\widetilde{\zeta} \in \widetilde{\mathcal{T}}_1$ such that $V_1(\widetilde{\zeta}, t_n) \leq V_1(\zeta, t_n)$. Hence, we can start with the construction of a new tree $\mathcal{T}_2$ with a wider control set $U_2 \supset U_1$ such that the nodes are constrained in the zones where the previous value function had the lowest values, *i.e.*

$$\min_{\widetilde{\mathcal{T}}_1^n} \widetilde{\zeta} \leq \zeta_j^n \leq \max_{\widetilde{\mathcal{T}}_1^n} \widetilde{\zeta}, \quad \forall \zeta_j^n \in \mathcal{T}_2^n,\ n \in \{N_{start}, \dots, N_t\}$$

where the minimum and the maximum are computed element-wise, as well the inequalities. The statistical pruning is applied starting from an arbitrary time $t_{N_{start}}$ since the first levels contain few nodes. We usually will fix $N_{start} = 3$. The entire procedure can be iterated doubling the number of controls in each step. Computed the tree $\widetilde{\mathcal{T}}_k$ at the k-th iteration, the subsequent tree $\mathcal{T}_{k+1}$ will satisfy the constraint

$$\min_{\widetilde{\mathcal{T}}_k^n} \widetilde{\zeta} \leq \zeta_j^n \leq \max_{\widetilde{\mathcal{T}}_k^n} \widetilde{\zeta}, \quad \forall \zeta_j^n \in \mathcal{T}_{k+1}^n,\ n \in \{N_{start}, \dots, N_t\}. \tag{5.2}$$

Since we neglect the nodes which do not satisfy (5.2), the problem can be regarded as a state-constrained problem where the constraint is given by the relation (5.2). We refer to [5] for more details about the coupling of the tree with state-constrained problems. The ratio ρ is fixed such that it still retains the optimal trajectory from the previous iteration. In this way we can ensure that the value function is not increasing during the iterative procedure. Therefore, we denote by $V_k^n(x)$ the value function obtained in the k-th iteration at the point x at time t_n. By construction, we can notice that the iterative value function at the initial time is non increasing, *i.e.*

$$V_{k+1}^0(x) \leq V_k^0(x), \quad \forall k \geq 0$$

and bounded from below since

$$V_k^n(x) \geq -TM_f - M_g, \quad \forall k \geq 0,\ n \in \{0, \dots, N_t\},\ x \in \mathcal{T}_k^n,$$

using the hypothesis (2.3). Hence, the iterative scheme is convergent and it can repeated until we reach a maximum number of iterations or it satisfies a stopping criterion. In Algorithm 3 the method is sketched.

Algorithm 3 Statistical pruning

```
1: Choose an initial condition $x_0$, a ratio $\rho$, a starting time $t_{N_{start}}$, a tolerance $tol$, a maximum number of
   iteration $k_{max}$ and a initial number of discrete controls $M$
2: Build a tree $\mathcal{T}_1$ with $M$ controls and compute the value function $V_1^n(x)$
3: while $res > tol$ and $k \leq k_{max}$ do
4:     $M := 2M - 1$
5:     Construct $\widetilde{\mathcal{T}}_k$ retaining a ratio $\rho$ of $\mathcal{T}_k$ with the lowest value function
6:     Construct $\mathcal{T}_{k+1}$ under the constraint (5.2)
7:     Compute the value function $V_{k+1}^n(x)$
8:     $res = |V_k^0(x_0) - V_{k+1}^0(x_0)|$
9:     $k = k + 1$
10: end while
```

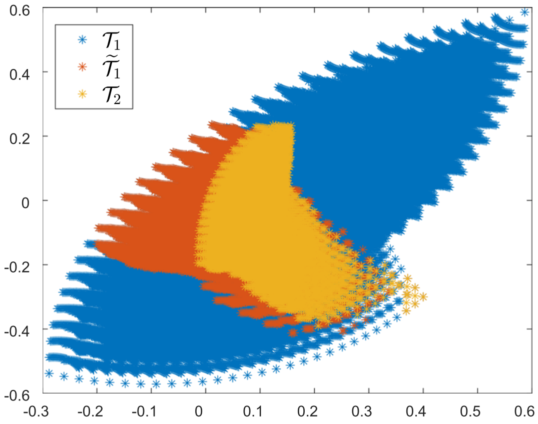

Fig. 1 Application of the statistical pruning to Van der Pol oscillator with $\rho = 0.3$

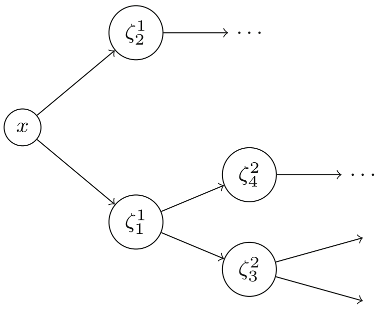

Fig. 2 Example of the tree $\overline{\mathcal{T}}_{2,n}$

In Fig. 1 we show an application of the statistical pruning under the Van der Pol dynamics:

$$\begin{cases} \dot{y_1}(t) = y_2(t), \quad t \in (0, T], \\ \dot{y_2}(t) = \omega(1 - y_1^2(t))y_2(t) - y_1(t) + u(t), \quad t \in (0, T], \\ (y_1(0), y_2(0)) = (0.4, -0.3), \end{cases}$$

where $\omega = 0.15$, $T = 1.4$ and $u : [0, T] \to [0, 1]$. Fixing a time step $\Delta t = 0.2$, we display the initial full tree $\mathcal{T}_1$ with discrete controls $\{0, 1\}$, its refinement $\widetilde{\mathcal{T}}_1$ with $\rho = 0.3$ and the new tree $\mathcal{T}_2$ with discrete controls $\{0, 0.5, 1\}$.

5.3 Monotone Control

In this section we restrict the admissible set of controls to monotone controls, e.g.

$$\overline{\mathcal{U}} = \{u : [0, T] \to U,\ u(\cdot) \text{ monotone in } [0, T]\}.$$

In economy different problems can be formulated as optimal control problems with monotone controls (e.g. adjustment theory of investment problems). Under this constraint, in [11] Barron proved that the value function is a generalized solution of a quasi-variational inequality and its numerical treatment has been investigated in a series of papers [9, 42]. We consider the non decreasing case without loss of generality. Let us introduce the notation which will be useful in this section. We define as $\mathcal{T}_{M,N}$ the tree obtained using M discrete controls and N time steps, while we denote as $\overline{\mathcal{T}}_{M,N}$ the tree constructed via monotone controls. In Fig. 2 we show the structure of the tree $\overline{\mathcal{T}}_{2,N}$. In this case we are using 2 discrete controls $u_1 < u_2$.

When we apply u_2, the corresponding subtree will have just one node for each level, since the control cannot decrease by hypothesis.

In this framework we have a great improvement in terms of the cardinality of the tree, as stated in the following proposition.

Proposition 1 *Given M discrete controls and N time steps, the cardinality of the tree based on monotone controls is given by*

$$\left|\overline{\mathcal{T}}_{M,N}\right| = \frac{(M+N)!}{M!N!} \tag{5.3}$$

Proof We will proceed by induction on the pair (M, N). It is easy to check that $\left|\overline{\mathcal{T}}_{1,N}\right| = N+1$ and $\left|\overline{\mathcal{T}}_{M,1}\right| = M+1$. Now let us suppose (5.3) holds for a pair (M, N). First, we are going to prove that the result holds for the pair $(M, N+1)$. Given the particular structure of the tree, we can write

$$\left|\overline{\mathcal{T}}_{M,N+1}\right| = \sum_{k=1}^{M} \left|\overline{\mathcal{T}}_{k,N}\right| + 1 = \frac{(M+N+1)! - M!(N+1)!}{M!(N+1)!} + 1 = \frac{(M+N+1)!}{M!(N+1)!},$$

obtaining the result. Afterwards, let us demonstrate it for the pair $(M+1, N)$. In this case we can split the tree in the following way

$$\begin{aligned}\left|\overline{\mathcal{T}}_{M+1,N}\right| &= \left|\overline{\mathcal{T}}_{M,N}\right| + \left|\overline{\mathcal{T}}_{M+1,N-1}\right| = \sum_{k=2}^{N} \left|\overline{\mathcal{T}}_{M,k}\right| + \left|\overline{\mathcal{T}}_{M+1,1}\right| \\ &= \frac{(M+N+1)! - (M+2)!N!}{(M+1)!N!} + M + 2 = \frac{(M+N+1)!}{(M+1)!N!}\end{aligned}$$

and this completes the proof. □

When the number of discrete controls M is fixed, the cardinality described in equation (5.3) scales as $O(N^M)$. Conversely, when the number of time steps N is fixed, the cardinality of the pruned tree scales as $O(M^N)$. Thus, we observe a polynomial order of magnitude in both cases, yielding an affordable algorithm for the computation of the optimal control.

5.4 Bilinear Control

Let us consider the following bilinear dynamical system:

$$\dot{y} = Ly + uy, \quad u \in U \subset \mathbb{R}. \tag{5.4}$$

Discretizing (5.4) via a semi-implicit scheme, we obtain

$$y^n = (I - \Delta t L)^{-1} y^{n-1}(1 + \Delta t u^{n-1}) = (I - \Delta t L)^{-n} y_0 \prod_{i=0}^{n-1} (1 + \Delta t u^i). \tag{5.5}$$

Let us consider now a new evolution $\tilde{y}^n$ of the discrete scheme at time t_n with controls $\{\tilde{u}^i\}_{i=0}^{n-1}$. Then the distance between the two dynamics is given by

$$\|y^n - \tilde{y}^n\| \le \|(I - \Delta t L)^{-n} y_0\| \left|\prod_{i=0}^{n-1}(1 + \Delta t u^i) - \prod_{i=0}^{n-1}(1 + \Delta t \tilde{u}^i)\right|. \tag{5.6}$$

Let us introduce a definition which will be useful in this section.

Definition 5.1 A discrete system with discrete controls satisfies the sum-based pruning property if for every pair of vectors $(u^0, \dots, u^{n-1})$ and $(\tilde{u}^0, \dots, \tilde{u}^{n-1})$ such that

$$\sum_{i=0}^{n-1} u^i = \sum_{i=0}^{n-1} \tilde{u}^i \tag{5.7}$$

the corresponding discrete solution y^n and $\tilde{y}^n$ satisfy the geometrical pruning rule, e.g. $\|y^n - \tilde{y}^n\| \le C\Delta t^2$.

This class of discrete systems benefits from an important improvement in terms of the cardinality of the corresponding tree, as stated in the following proposition. For the proof we refer to Proposition 3.12 in [48].

Proposition 2 *The cardinality of the tree based on a system with M discrete controls and N time steps satisfying the sum-based pruning property is at most* $\frac{N(N-1)}{2}(M-1) + N + 1$.

The discrete dynamics (5.5) with two discrete controls belongs to the class of the system verifying the sum-based pruning property as stated in the next proposition.

Proposition 3 *The discrete system* (5.5) *satisfies the sum-based pruning with* 2 *discrete controls. Hence, the cardinality of the corresponding tree is at most* $\frac{N(N+1)}{2} + 1$.

Proof Let us consider two pair of vectors $(u^0, \dots, u^{n-1})$ and $(\tilde{u}^0, \dots, \tilde{u}^{n-1})$ verifying the sum-based pruning property (5.7). Since we are considering two discrete control $(u_1, u_2) \in U \times U$, the upper bound for distance (5.6) between the two corresponding dynamics becomes

$$\|(I - \Delta t L)^{-n} y_0\| \left|(1 + \Delta t u_1)^{k_1}(1 + \Delta t u_2)^{n-k_1} - (1 + \Delta t u_1)^{k_2}(1 + \Delta t u_2)^{n-k_2}\right|$$

with $k_1, k_2 \in \{0, \dots, n\}$. By property (5.7) we immediately see that either $k_1 = k_2$ or $u_1 = u_2$, which implies that the two corresponding solutions coincide. □

It should be noted that, according to Proposition 3, the cardinality of the tree with two discrete controls is reduced from $O(2^{N_t})$ to $O(N_t^2)$. In this scenario, the tree can be constructed directly based on this structure without the need for any pruning criteria. Constructing the tree with two discrete controls can serve as an efficient and cost-effective method for obtaining information about the full-dimensional system. Once the basis is constructed and the system is projected onto the lower-dimensional space, a higher number of discrete controls can subsequently be considered.

Remark 5.1 It is important to emphasize that, in the last two cases (monotone and bilinear), the cardinality of the tree pruned using the proposed techniques scales as $O(N_t^M)$. This approach is particularly effective when the number of discrete controls is relatively small (e.g., less than 10). However, if the problem requires greater refinement in the control set, statistical pruning is a more suitable approach.

6 An Error Bound for the Multilinear HJB-POD-DEIM Algorithm

The aim of this section is to derive an error estimate for the approximation of value function with the HO-POD-DEIM algorithm applied to the tree structure. The main reference of this section is [14], where the authors obtain a state space error bounds for the solutions of the reduced systems via a POD-DEIM approach and the application of an implicit scheme for the time integration. Following their proof, we are going to extend the result to semi-implicit schemes in our multilinear setting.

First of all, we consider the vectorized form of dynamical system (3.3), whose semi-implicit discretization with stepsize Δt and discrete controls $\{u^j\}_{j=0}^{N_t-1}$ reads

$$\frac{y^j - y^{j-1}}{\Delta t} = Ly^j + \boldsymbol{f}(y^{j-1}, u^{j-1}, t^{j-1}), \qquad j = 1, \dots, N_t. \tag{6.1}$$

Taking into account the basis in vector form

$$V_Y = V_d \otimes \cdots \otimes V_1, \quad V_F = \boldsymbol{\Phi_d} \otimes \cdots \otimes \boldsymbol{\Phi_1},$$
$$\mathbb{P} = V_F \left((\mathbf{P_d} \otimes \cdots \otimes \mathbf{P_1})^\top V_F \right)^{-1} (\mathbf{P_d} \otimes \cdots \otimes \mathbf{P_1})^\top,$$

the vectorized form of the semi-implicit scheme for the reduced dynamics (3.16) reads

$$\frac{\hat{y}^j - \hat{y}^{j-1}}{\Delta t} = V_Y^\top L\, V_Y \hat{y}^j + V_Y^\top \mathbb{P} \boldsymbol{f}(V_Y \hat{y}^{j-1}, u^{j-1}, t^{j-1}), \qquad j = 1, \dots, N_t. \tag{6.2}$$

Due to vectorization, it is feasible to apply existing results from the literature concerning dynamical systems in vectorized form. It is important to emphasize that this approach is merely an analytical technique used to establish the error bound for the approximation error. The superior performance achieved by the multilinear form has been discussed in previous sections and will be evident in the numerical experiments.

Our aim is to prove an error estimate between full order scheme (6.1) and reduced one (6.2).

For this purpose we introduce the logarithmic norm of matrix $A \in \mathbb{C}^{n\times n}$ defined as

$$\mu(A) = \sup_{x \in \mathbb{C}^n \setminus \{0\}} \frac{Re(< Ax, x >)}{\|x\|^2}, \tag{6.3}$$

where $Re(z)$ refers to the real part of $z \in \mathbb{C}$. The logarithmic norm plays an important role for the stability analysis for continuous and discrete linear dynamical systems. Indeed, it is possible to prove that $\|e^{tA}\| \leq e^{t\mu(A)}\ \forall t \geq 0$ (see e.g. [51]) and by this inequality we can state that the dynamical system is stable if $\mu(A) \leq 0$. The definition of this norm will be fundamental in the treatment of the implicit part of the scheme, while the Lipschitz-continuity of $\mathbf{f}$ will be employed for the estimation of the explicit part. In the following proposition we prove that the error between the full order model (6.1) and the lifted reduced order model (6.2) depends on the accuracy of the HO-POD and HO-DEIM basis. The proof can be found in Appendix A.

Proposition 6.1 *Given $\{y^k\}_{k=0}^{N_t}$ the solution of the (6.1) and $\{\hat{y}^k\}_{k=0}^{N_t}$ solution of (6.2) with controls $\{u^j\}_{j=0}^{N_t-1}$ and time step Δt satisfying $\Delta t\, \mu(V_Y^\top L V_Y) < 1$, then*

$$\sum_{k=0}^{N_t} |y^k - V_Y \hat{y}^k|^2 \leq C(T) \left(\mathcal{E}_y + \mathcal{E}_f\right) \tag{6.4}$$

with

$$\mathcal{E}_y = \sum_{j=0}^{N_t} |y^j - V_Y V_Y^\top y^j|^2, \quad \mathcal{E}_f = \sum_{j=0}^{N_t-1} |\boldsymbol{f}(y^j, t^j, u^j) - V_F V_F^\top \boldsymbol{f}(y^j, t^j, u^j)|^2.$$

Finally, we are ready to prove a convergence result for the continuous value function $v(x,t)$, solution of the HJB Eq. (2.8), and the discrete value function solution $\{\widehat{V}^n(\widehat{x})\}_n$, solution of the scheme (3.17). For this purpose, let us define the continuous version of the DDP for the full model

$$\begin{cases} V(x,s) = \min\limits_{u\in U}\{V(x+(t_{n+1}-s)f(x,u,s),t_{n+1}) + (t_{n+1}-s)\,L(x,u,s)\}, \\ V(x,T) = g(x), \quad x\in\mathbb{R}^d, s\in[t_n,t_{n+1}), \end{cases} \tag{6.5}$$

and its reduced version which reads:

$$\begin{cases} \widehat{V}(\widehat{x},s) = \min\limits_{u\in U}\{\widehat{V}(\widehat{x}+(t_{n+1}-s)f^\ell(x^\ell,u,s),t_{n+1}) + +(t_{n+1}-s)\,\widehat{L}(\widehat{x},u,s)\}, \\ \widehat{V}(\widehat{x},T) = \widehat{g}(\widehat{x}), \qquad \widehat{x}\in\mathbb{R}^\ell, s\in[t_n,t_{n+1}). \end{cases} \tag{6.6}$$

Given the exact value function $v(x,s)$ and its continuous reduced approximation $\widehat{V}(V_Y x,s)$, the following theorem provides an error estimate for the approximation of the HJB equation by the HO-POD-DEIM approach. The assumptions and the main procedure for the following result can be found in Theorem 5.1 in [7].

Theorem 6.1 *Given $v(x,s)$ the solution of the HJB Eq. (2.8) and its reduced approximation $\widehat{V}(V_Y x,s)$ solution of the scheme (6.6), the following estimate holds*

$$|v(x,s)-\widehat{V}(V_Y x,s)| \le C(T)\left(\Delta t + \mathcal{E}_y + \mathcal{E}_f\right). \tag{6.7}$$

Proof The proof follows closely the procedure adopted for Theorem 5.1 in [7]. The only difference arises in the estimation of the projection error between the FOM and the lifted ROM solutions and in this case we apply Proposition 6.1 to obtain the result. □

7 Numerical Tests

In this section we test the proposed technique in different frameworks. In the first numerical test we consider a bilinear advection–diffusion equation, comparing the vector and matrix cases for the construction of the reduced basis. In this numerical test we employ the geometrical pruning strategy in the online stage, taking into account the results obtained in Sect. 5.4 for bilinear optimal control problems for the offline stage. Moreover, a comparison between the geometrical pruning and the monotone control is presented. The second test is devoted to a nonlinear reaction-diffusion PDE where we show the efficiency of the statistical pruning coupled with the MOR technique. Finally, in the third test we consider a more challenging problem: the control of the 3D viscous Burgers' equations, where the tree structure algorithm is coupled with the geometrical pruning. We use this final example to indicate the power of the proposed algorithm in terms of CPU time and memory requirements with respect to the vector construction of the problem. The numerical tests are performed on a Dell XPS 13 with Intel Core i7, 2.8GHz and 16GB RAM. The codes are written in Matlab R2022a.

7.1 Test 1: Advection–Diffusion Equation

In the first numerical test we conside the following bilinear advection–diffusion equation:

$$\begin{cases} \partial_s y + (c_1,c_2)\cdot\nabla y = \sigma\Delta y + yu(s) & (x,s)\in\Omega\times[0,T], \\ y(x,s)=0 & (x,s)\in\partial\Omega\times[0,T], \\ y(x,0)=y_0(x) & x\in\Omega, \end{cases} \tag{7.1}$$

Table 1 Dim. of basis for POD and HO-POD varying the number of the grid points per dimension n with $\tau = 10^{-4}$, $\Delta t = 0.05$ and two discrete controls

n	101	121	141	161	181	201
POD	7	7	7	7	8	8
HO-POD	(3, 7)	(3, 7)	(3, 7)	(3, 7)	(3, 8)	(3, 8)

with

$$u \in \mathcal{U} = \{u : [0, T] \to U, u \, measurable\}.$$

The aim of the optimal control is to drive the solution to the equilibrium $\overline{y}(x) \equiv 0$ and to this end we introduce the following cost functional:

$$J_{y_0,t}(u) = \int_t^T \left(\int_\Omega |y(x,s)|^2 dx + |u(s)|^2 \right) ds + \int_\Omega |y(x,T)|^2 dx.$$

Since we are considering a bilinear optimal control problem, we can benefit of the results presented in Sect. 5.4, e.g. discretizing the control set with two discrete controls, the total cardinality of the tree is order $O(N_t^2)$. We fix $T = 1$, $U = [-3, -1]$, $y_0(x) = \max(2 - \|x\|^2, 0)$, $\Omega = [-5, 5]^2$, $\widehat{\Delta t} = \Delta t = 0.05$, $\widehat{U} = \{-3, -1\}$, $c_1 = 0.5$, $c_2 = 0$ and $\sigma = 0$. Later on we will discuss the behaviour of the algorithm considering different choices for the coefficients c_1, c_2 and σ. Furthermore, we impose the truncation tolerance $\tau = 10^{-4}$ in the criterion (3.13) for both methods to obtain the same projection error. In this setting the cardinality of the tree is 3321. In Table 1 we show the dimension of the basis varying the parameter n, where n is the number of the grid points in each direction. Since the system is driven along an axes, the HO-POD procedure requires more basis in one direction with respect to the other. We note that the maximum of the dimensions of the HO-POD basis is equal to the number of POD basis for any choice of n.

In the top left panel of Fig. 3 we compare the CPU time for the offline phase for the POD and HO-POD algorithms. As stated previously, HO-POD requires less storage and enables to treat with very high dimensional problems. In particular, we note a difference of almost two orders of magnitude between the POD and HO-POD offline stages for $n = 201$. In the top right and bottom left panels of Fig. 3 a comparison of the computational times and cardinality of the pruned trees varying the number of discrete controls for the online phase and fixing $n = 161$ is presented. The HO-POD is again performing better than the POD algorithm since the geometrical pruning turns to be more efficient in the HO-POD setting. Indeed, fixing $M = 7$ discrete controls, the cardinality of the HO-POD tree reaches almost order 10^5, against the order $\approx 10^7$ for the POD tree and 10^{16} for the unpruned tree. Given that we are exploring various projection techniques, the resulting reduced dynamics differ, which may account for the variation in the cardinality of the pruned trees across the different reduced settings. In the bottom right panel of Fig. 3, we present the optimal control computed using both techniques, with the number of discrete controls fixed at four. The behavior of the control signals is similar, except for a shift. The total cost of the controlled HO-POD trajectory is 2.0077 compared to 2.1149 for the controlled POD dynamics, indicating superior performance in cost minimization. Furthermore, the controls exhibit a monotone behavior, suggesting the appropriateness of a monotone control strategy, as demonstrated in the following subsection.

The optimal trajectory computed via HO-POD with 7 discrete controls for different time instances is displayed in Fig. 4, noting that the solution is getting closer to the stationary solution.

Finally, Table 2 shows the different CPU times for the offline phase for the two methods.

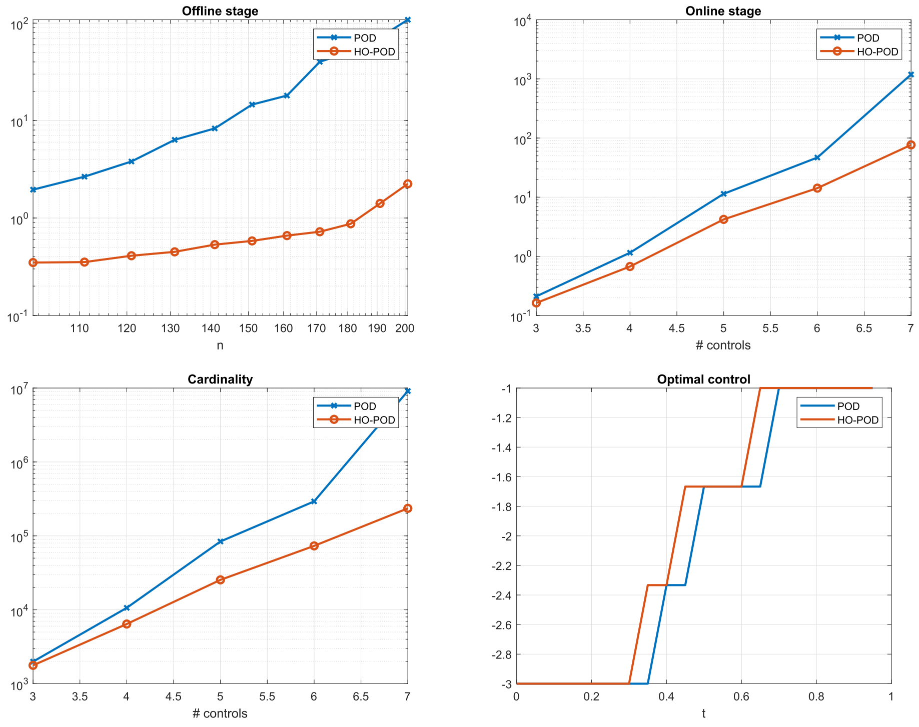

Fig. 3 Test 1: CPU time for the offline stage (top left), CPU time for the online stage (top right), cardinality of the pruned tree (bottom left) and optimal controls (bottom right) for POD and HO-POD techniques

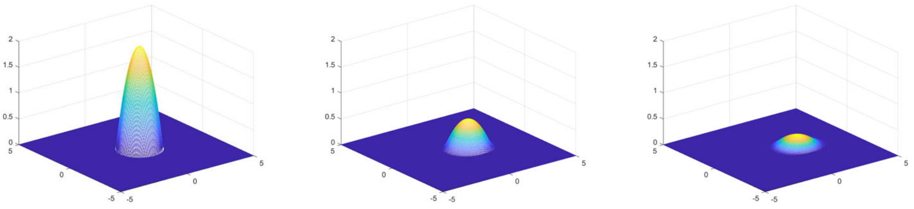

Fig. 4 Test 1: Optimal trajectory for $t = 0$ (left), $t = 1$ (central) and $t = 2$ (right) for HO-POD and $n = 161$

Table 2 Offline CPU times for POD and HO-POD varying the coefficient (c_1, c_2, σ) with $n = 601$ and $\widehat{\Delta t} = 0.05$

(c_1, c_2, σ)	POD	HO-POD
(1, 0, 0)	17.78s	0.67s
(1, 1, 0)	18 s	1.77s
(0, 0, 1)	16.9s	0.84s
(1, 1, 1)	34 s	1.73s

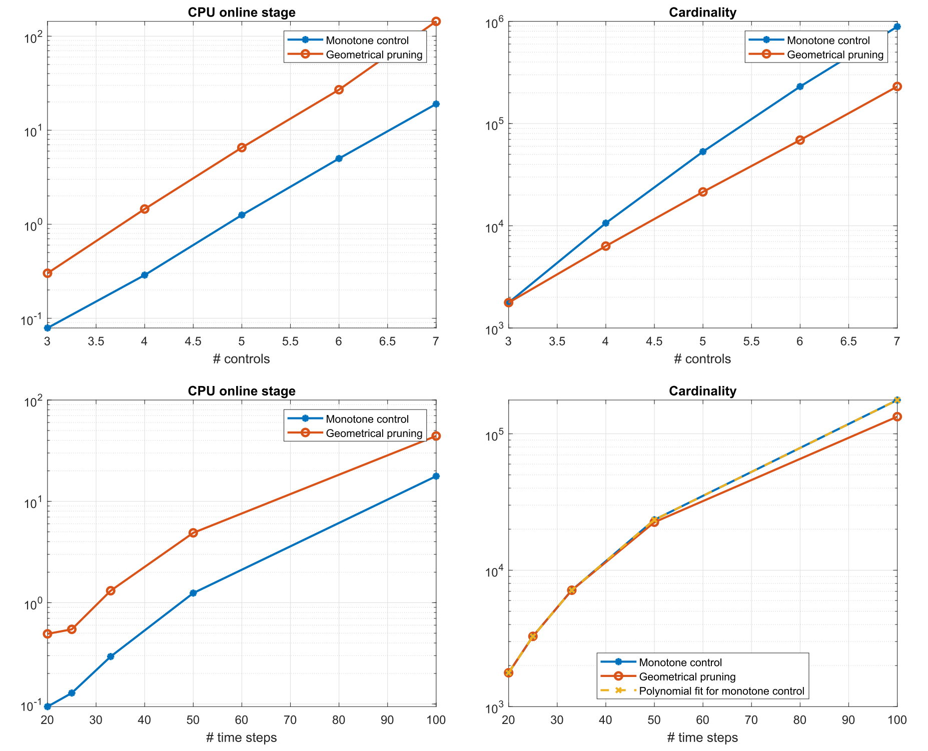

Fig. 5 Test 1: Comparison with the monotone control. Top: CPU time for the online phase (left) and cardinality of the tree (right) fixing 21 time steps. Bottom: CPU time for the online phase (left) and cardinality of the tree (right) fixing 3 discrete controls

First of all, we notice that HO-POD is faster in all cases, but we obtain a particular speed-up in presence of a one-direction convection, since the construction of the basis operates separately in each direction.

Comparison with the monotone control

Let us now compare the performances of geometrical pruning with those obtained by imposing monotone control in the HO-POD setting. In the previous simulation, it was observed that the control signal exhibits a monotone behavior, making it a suitable setting for investigating the monotone control framework. We consider two test cases:

1. The number of time steps $N_t = 21$ is fixed, and the number of discrete controls M is varied within the set $\{3, 4, 5, 6, 7\}$;
2. The number of discrete controls $M = 3$ is fixed, and the time step Δt is varied within the set $\{0.05, 0.04, 0.03, 0.02, 0.01\}$.

The results are displayed in Fig. 5. In both cases, we observe that the online phase using monotone control is faster than that using geometrical pruning, with both approaches exhibiting similar growth rates. However, the cardinality of the tree pruned via geometrical pruning is lower, indicating that geometrical pruning more effectively reduces the cardinality of the tree, albeit at the cost of an expensive node closeness check. Conversely, the monotone approach fixes the tree structure, resulting in a more efficient algorithm. In the bottom left panel of Fig. 5, a third-order polynomial fit is also presented, demonstrating that the cardinality of the

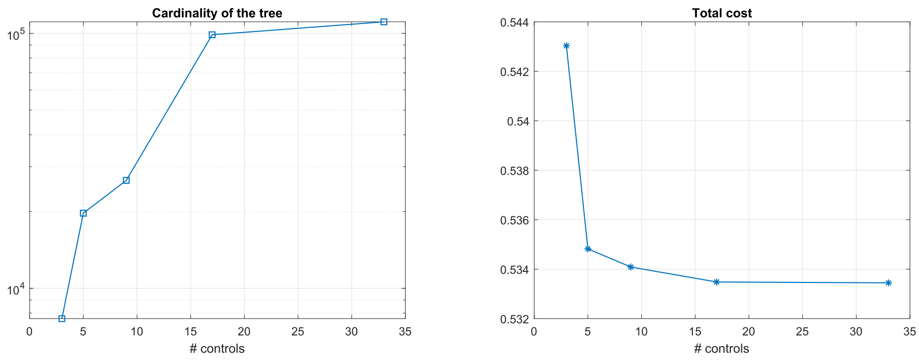

Fig. 6 Test 2: Cardinality of the tree in logarithmic scale (left) and total cost (right) for $\Delta t = 0.1$ and varying the number of controls

pruned tree grows as N_t^M, where $M = 3$ represents the number of discrete controls. Finally, the total costs of the two approaches are very similar and thus are not reported.

7.2 Test 2: Allen–Cahn Equation

We consider the following nonlinear PDE with homogeneous Neumann boundary conditions:

$$\begin{cases} \partial_s y = \sigma \Delta y + y\left(1 - y^2\right) + y_0(x)u(s) & (x, s) \in \Omega \times [0, T], \\ \partial_n y(x, s) = 0 & (x, s) \in \partial\Omega \times [0, T], \\ y(x, 0) = y_0(x) & x \in [-1, 1]^2. \end{cases} \tag{7.2}$$

Our aim is to steer the solution to the unstable equilibrium $\overline{y} \equiv 0$ minimizing the following cost functional

$$J_{y_0,t}(u) = \int_t^T \left(\int_\Omega |y(x, s)|^2 dx + \gamma |u(s)|^2 \right) ds + \int_\Omega |y(x, T)|^2 dx.$$

We fix $T = 1$, $\gamma = 0.01$, $\sigma = 0.1$, $\Omega = [-1, 1]^2$ $U = [-2, 0]$ and $y_0(x) = 2 + \cos(2\pi x_1)\cos(2\pi x_2)$. Furthermore, we set $\widehat{\Delta t} = \Delta t = 0.1$, $\tau = 10^{-3}$ and we discretize the domain $[-1, 1]^2$ with 601 equidistant points, obtaining a grid of 361201 points. In the offline phase we consider 2 discrete controls ($\widehat{U} = \{-2, 0\}$) and we construct a rough tree with 2047 nodes. Since the problem is not linear, we apply the HO-POD-DEIM strategy and the dimensions of the basis turns to be $k_1 = k_2 = p_1 = p_2 = 5$, hence the low dimension solution lives in $\mathbb{R}^{5\times 5}$. The offline stage took 174 seconds. In this example the dynamical system is nonlinear and we do not have any a priori estimate for the introduced pruning criteria. Hence, we are going to apply the statistical pruning discussed in Sect. 5.2. We fix the ratio $\rho = 0.2$ and we iterative the statistical pruning strategy explained in Algorithm 3 with a stopping tolerance $tol = 10^{-4}$. The number of discrete controls in the online phase ranges within the set $\{3, 5, 9, 17, 33\}$, and these controls are uniformly spaced over the interval $[-2, 0]$. In the left panel of Fig. 6 we show the cardinality of the low dimensional tree in logarithm scale. We recall that the cardinality of the full tree would be $O(M^{11})$, where M is the number of discrete controls, reaching an order of $\approx 10^{17}$ in the case of 33 controls. The application of the statistical pruning achieves a great improvement in terms of memory storage, gaining almost 12 orders of magnitudes with respect to the full tree with 33 discrete

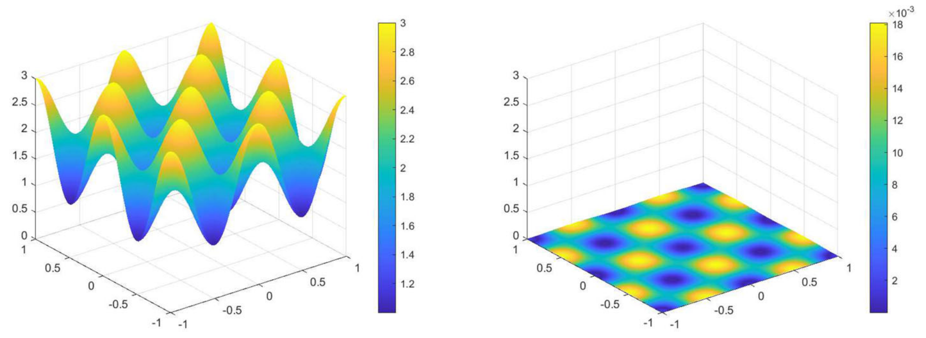

Fig. 7 Test 2: Optimal trajectory at $t = 0$ (left) and $t = 0.5$ (right) for $\Delta t = 0.1$ and $M = 33$

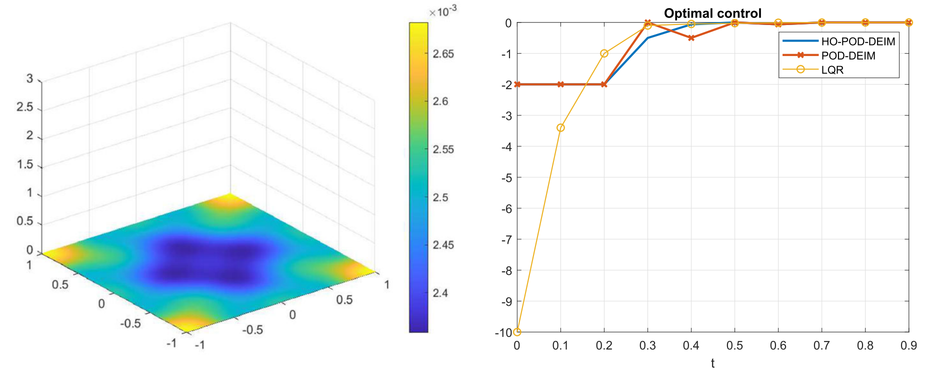

Fig. 8 Test 2: Optimal trajectory at $t = 1$ (left) for $\Delta t = 0.1$ and $M = 33$. Comparison of the control signals between the LQR tool and the TSA based on HO-POD-DEIM and POD-DEIM (right)

controls. The total cost varying the number of controls is displayed in the left panel of Fig. 6. The cost functional shows a decreasing behaviour as expected and the algorithms stops with 33 controls since the stopping rule has been satisfied. In Fig. 7 and in the left panel of Fig. 8 the optimal trajectories at time instances $t \in \{0, 0.5, 1\}$ are displayed. We note that the control signal is driving the solution to the unstable equilibrium $\tilde{y} \equiv 0$. We aim to demonstrate the efficiency of the TSA, based on the HO-POD-DEIM method, by comparing it to the POD approach and the widely used Linear Quadratic Regulator (LQR). The configuration of the offline stage for the POD approach is identical to that used in the HO-POD technique. Since we are addressing a finite horizon optimal control problem, it is necessary to solve a Differential Riccati Equation (DRE). The matrix structure of the DRE is preserved using matrix generalizations of classic BDF methods (refer to [18] for further details). For a meaningful comparison with the first-order scheme used in our methods, we employ the first-order BDF method, maintaining the same time step used in the construction of the tree structure. We observe that the POD-DEIM and HO-POD-DEIM controls coincide at the initial and final time steps, showing some differences in the middle of the time interval. Conversely, the LQR control takes higher values in modulus at the beginning since it does not impose any constraints on the control set. We now aim to compare the three approaches with respect to total costs for varying initial conditions. Consider the family of initial conditions defined as $y_0(x, \alpha) = \alpha(2 + \cos(2\pi x_1)\cos(2\pi x_2))$. In Table 3, we present the total costs

Table 3 Comparison of the total costs for the LQR tool and the TSA based on HO-POD-DEIM and POD-DEIM for different initial conditions $y_0(x, \alpha)$

Method	$\alpha = 1$	$\alpha = 10^{-2}$	$\alpha = 10^{-4}$
HO-POD-DEIM	0.5334	3.71e−3	8.40e−5
POD-DEIM	0.5456	5.22e−3	2.45e−4
LQR	0.7863	7.53e−3	6.47e−5

for the three methods, with α varying over the set $\{1, 10^{-2}, 10^{-4}\}$. In the first two cases, the HO-POD-DEIM approach achieves the lowest cost, with POD-DEIM yielding a slightly higher, yet comparable, cost. In contrast, the LQR method results in the highest cost for these cases. However, for the initial condition with the smallest magnitude, LQR produces the lowest cost. This indicates that incorporating nonlinear terms in the control strategy leads to superior performance for initial conditions of moderate magnitude, whereas for initial conditions closer to the origin, LQR serves as an effective controller.

7.3 Test 3: 3D Viscous Burgers' Equation

Here we consider the nonlinear 3D viscous Burgers' equation (see, e.g., [23]) given by

$$\begin{cases} \partial_t y_1 &= \frac{1}{r}\Delta y_1 - \underline{y} \cdot \nabla y_1 + y_1 u \\ \partial_t y_2 &= \frac{1}{r}\Delta y_2 - \underline{y} \cdot \nabla y_2 + y_2 u, \\ \partial_t y_3 &= \frac{1}{r}\Delta y_3 - \underline{y} \cdot \nabla y_3 + y_3 u, \end{cases} \tag{7.3}$$

where $y_1(x_1, x_2, x_3, t)$, $y_2(x_1, x_2, x_3, t)$ and $y_3(x_1, x_2, x_3, t)$ are the three velocities to be determined, with $x = (x_1, x_2, x_3) \in [0, 1]^3$, $t \in [0, 1]$ and the Reynold's number $r = 100$. Furthermore, the system is subject to homogeneous Dirichlet boundary conditions and initial states

$$y_1(x, y, z, 0) = \frac{1}{10}\sin(2\pi x_1)\sin(2\pi x_2)\cos(2\pi x_3)$$
$$y_2(x, y, z, 0) = \frac{1}{10}\sin(2\pi x_1)\cos(2\pi x_2)\sin(2\pi x_3)$$
$$y_3(x, y, z, 0) = \frac{1}{10}\cos(2\pi x_1)\sin(2\pi x_2)\sin(2\pi x_3).$$

A finite difference space discretization in the cube yields a system of ODEs of the form (3.7), with nonlinear functions given by

$$\mathcal{F}_i(\mathcal{D}_i(\boldsymbol{\mathcal{Y}}_i), \boldsymbol{\mathcal{Y}}_1, \boldsymbol{\mathcal{Y}}_2, \boldsymbol{\mathcal{Y}}_3, t) = \sum_{k=1}^{3} (\boldsymbol{\mathcal{Y}}_i \times_k B_{ki}) \circ \boldsymbol{\mathcal{Y}}_k,$$

for $i = 1, 2, 3$, where $B_{1i} \in \mathbb{R}^{n\times n}$, $B_{2i} \in \mathbb{R}^{n\times n}$ and $B_{3i} \in \mathbb{R}^{n\times n}$ contain the coefficients for a first order centered difference space discretization in the x_1-, x_2- and x_3- directions respectively, and n is the dimension of the discretized tensor in each spatial direction. For a more detailed discussion on the space discretization and HO-POD-DEIM model reduction of *systems of ODEs* in array form, we point the reader to [29], as well as the companion manuscript [22].

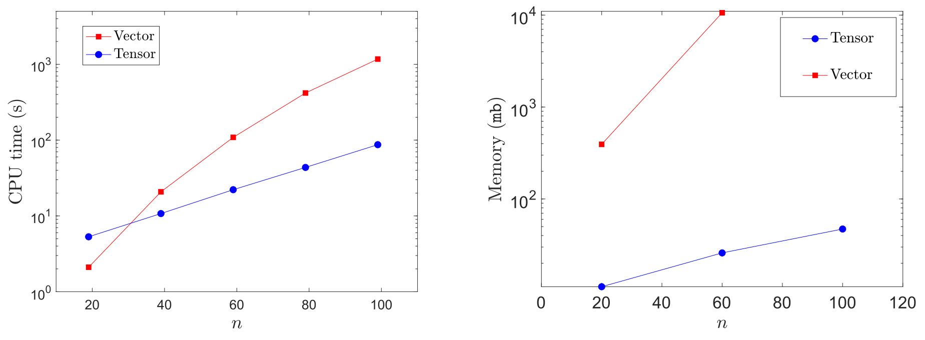

Fig. 9 Test: CPU time (left) and memory requirements (right) for both methods applied to (7.3)

We consider the following cost functional

$$J_{y_0,t}(u) = \int_t^T \left(\int_\Omega \sum_{i=1}^{3} |y_i(x,s)|^2 dx + \frac{1}{10} |u(s)|^2 \right) ds + \int_\Omega \sum_{i=1}^{3} |y_i(x,T)|^2 dx. \quad (7.4)$$

The control $u(t)$ will be taken in the following admissible set of controls

$$\mathcal{U} = \{u : [0,T] \to [-2,0]\}.$$

We therefore construct one tree for the control u containing the approximate solution of each of the three equations at its nodes. Constructing and storing this tree of course leads to extremely demanding memory requirements and computational effort to construct the approximations spaces for the reduced models.

We therefore use this experiment to illustrate the massive computational gain of the HO-POD-DEIM method, in combination with the snapshot selection algorithm and the low-rank storage algorithm. We first investigate the computational load required in the offline phase by the HO-POD-DEIM method as well as standard POD-DEIM applied to the system (7.3) discretized in vector form. For the vectorized system we also apply a semi-implicit Euler time discretization to each of the three equations, and each linear system is solved using the Matlab function `pcg` preconditioned with an incomplete Cholesky factorization with a drop tolerance of 10^{-4}.

Below we illustrate the computational load both in terms of CPU time and memory requirements. On the left of Fig. 9 we plot the computational time required by both methods to construct the full-dimensional tree, with $N_t = 10$ and two controls, whose nodal values are used to construct either the HO-POD-DEIM basis or the standard POD-DEIM basis. To construct the tree, all nodal values from the previous time level need to be stored. To this end, one of the computational bottlenecks in the construction of the reduced model is memory requirements. Consequently, we further illustrate the power of the proposed algorithm on the right of Fig. 9, where we plot the maximum memory required (in `mb`) at any point in the offline phase of the respective algorithms. The plot indicates a massive difference in memory requirements, mainly related to the low-rank basis construction used in the HO-POD-DEIM algorithm, as well as the low-rank memory allocation method discussed in Sect. 4.2. Furthermore we notice, that no data points are plotted in the vector case for $n > 60$, as this is where the computer ran out of its available computational memory. Both plots are with respect to increasing n and we select $\tau = 10^{-2}$ and $\kappa = 20$ a priori.

Table 4 Dim. of basis and the average relative error compared to the full order model with dimension $n = 60$ and $\tau = 10^{-2}$

y	k_1	k_2	k_3	p_1	p_2	p_3	ERROR
y_1	6	11	12	10	18	19	$2 \cdot 10^{-2}$
y_2	6	15	13	6	20	17	$3 \cdot 10^{-2}$
y_3	6	12	13	5	18	18	$2 \cdot 10^{-2}$

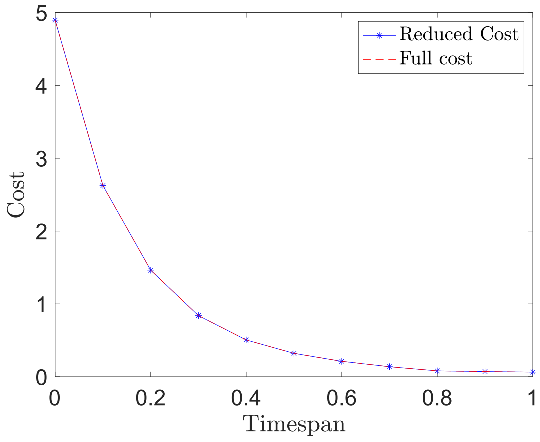

Fig. 10 Cost functional for the optimal control

In what follows we consider the reduced model constructed by the HO-POD-DEIM method and investigate the efficiency of the reduction based on an offline phase with $\widehat{\Delta t} = 0.1$ and $\widehat{U} = \{-2, 0\}$. In Table 4 we indicate the dimensions of the reduced approximation spaces determined to comply with $\tau = 10^{-2}$ and $n = 60$. In the online phase, using $\Delta t = 0.1$, three discrete controls $\{-2, -1, 0\}$, and the standard geometric pruning technique, the reduced model with dimensions specified in Table 4 required only 19 s, compared to the 361 s needed by the full-order model.

Finally, we also plot the cost functional in Fig. 10 for both the full and reduced order models as well as the optimal trajectories (unfolded along the first mode) for all three equations at $t = 0$, $t = 0.5$ and $t = 1$ in Fig. 11. Both these plots, indicate the convergence to the equilibrium of the reduced model. We note that there is a visual superposition of the two curves of the cost functional, demonstrating the effectiveness of the proposed methodology for determining the optimal trajectory.

8 Conclusions

In this paper we have introduced a new algorithm for approximating optimal feedback controls related to optimal control problems driven by evolutionary partial differential equations. The new algorithm is based on a tree structure to avoid the construction of a grid in the solution of the HJB equations, and exploits the compact representation of the dynamical systems based on tensor notations via a higher-order model reduction approach. We have shown how the algorithm can be constructed for general nonlinear control problems and given some crucial hints on its implementation. Furthermore, we have studied the existing pruning techniques for reducing the cardinality of the constructed tree, and introduced a new *statistical pruning* technique for a further reduction in the cardinality of the tree. To guarantee the convergence

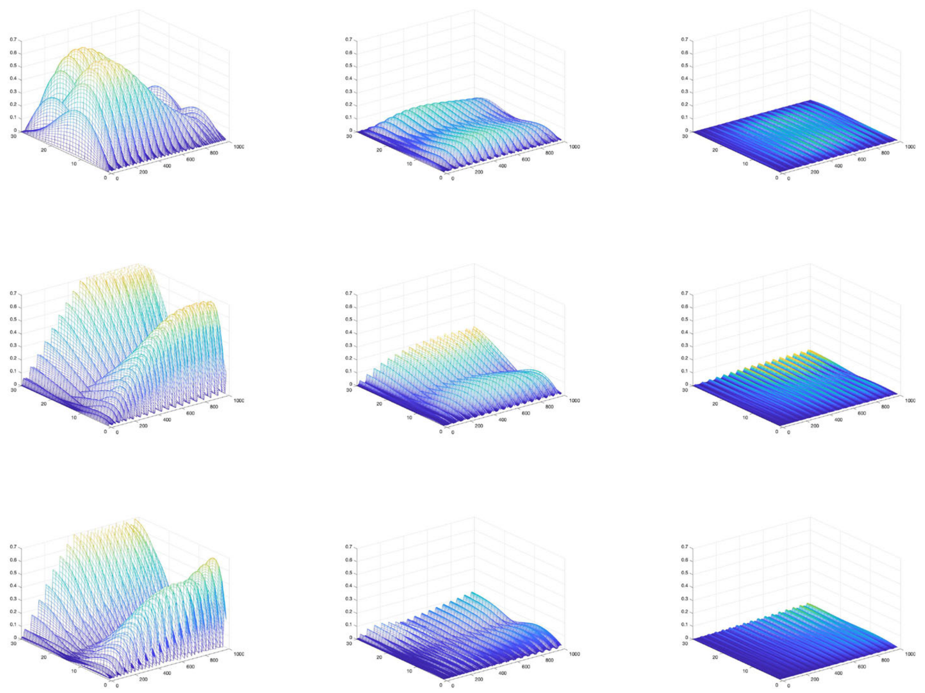

Fig. 11 All three controlled trajectories from top to bottom respectively, unfolded along the first mode. We plot $t = 0$ (left), $t = 0.5$ (middle) and $t = 1$ (right)

of the method, we derived an error estimate depending on the time step and on the accuracy of the HO-POD-DEIM basis. Finally, numerical tests on a number of challenging benchmark problem have been discussed, indicating the power of the method, with large savings both in terms of computational time and, especially, memory. We believe that these promising numerical results brings us one step closer to the application of DP in challenging, industrial settings.

To this end, we plan to, in the near future, explore more challenging industrial problems where the combination of the compact tensor representation of the problem and the tree structure algorithm can give a competitive advantage to DP for feedback control problems over possible competitors.

Appendix A: Proof of Proposition 6.1

Proof For the sake of simplicity we are going to define $\hat{L} = V_Y^\top L V_Y$, $f_k = \boldsymbol{f}(y^k, t^k, u^k)$, $\hat{f}_k = V_Y^\top \mathbb{P}\boldsymbol{f}(V_Y \hat{y}^k, u^k, t^k)$ and $f_{k,V} = V_Y^\top \mathbb{P}\boldsymbol{f}(V_Y V_Y^\top y^k, u^k, t^k)$, where we are considering the same control sequence $\{u_k\}_{k=0}^{N_t-1}$.

We consider the error at time t_j between the full model and the lifted reduced model as

$$E_j = y^j - V_Y \hat{y}^j$$

and we rewrite it as a sum of two quantities

$$E_j = \rho_j + \theta_j$$

where

$$\rho_j = y^j - V_Y V_Y^\top y^j, \quad \theta_j = V_Y V_Y^\top y^j - V_Y \hat{y}^j.$$

Multiplying (6.1) by $V_Y^\top$ and adding and subtracting $\hat{L} V_Y^\top y^j + f_{j-1,V}$ we get

$$V_Y^\top \frac{y^j - y^{j-1}}{\Delta t} = \hat{L} V_Y^\top y^j + f_{j-1,V} + \hat{R}_j,$$

with

$$\hat{R}_j = V_Y^\top L y^j + V_Y^\top f_{j-1} - \hat{L} V_Y^\top y^j - f_{j-1,V}.$$

Defining $\hat{\theta}_j = V_Y^\top \theta_j$, we obtain

$$\frac{\hat{\theta}_j - \hat{\theta}_{j-1}}{\Delta t} = V_Y^\top \frac{y^j - y^{j-1}}{\Delta t} - \frac{\hat{y}^j - \hat{y}^{j-1}}{\Delta t} = \hat{L} V_Y^\top y^j + f_{j-1,V} + \hat{R}_j - \hat{L}\hat{y}^j - \hat{f}_{j-1}.$$

Since $\left\langle \hat{\theta}_j, \hat{\theta}_{j-1} \right\rangle \leq \|\hat{\theta}_j\| \|\hat{\theta}_{j-1}\|$, we get

$$\begin{aligned}
\frac{\|\hat{\theta}_j\| - \|\hat{\theta}_{j-1}\|}{\Delta t} &\leq \frac{1}{\|\hat{\theta}_j\|} \left\langle \hat{\theta}_j, \frac{\hat{\theta}_j - \hat{\theta}_{j-1}}{\Delta t} \right\rangle \\
&= \frac{1}{\|\hat{\theta}_j\|} \left(\left\langle \hat{\theta}_j, \hat{L}\left(V_Y^\top y^j - \hat{y}^j \right) \right\rangle + \left\langle \hat{\theta}_j, f_{j-1,V} - \hat{f}_{j-1} + \hat{R}_j \right\rangle \right) \\
&\leq \mu(\hat{L}) \|\hat{\theta}_j\| + \gamma \|\hat{\theta}_{j-1}\| + \|\hat{R}_j\|,
\end{aligned}$$

where $\gamma = L_{\mathbf{f}} \| V_Y^\top \mathbb{P} \|$ and we used the definition of logarithmic norm (6.3) and the Lipschitz-continuity of the function $\mathbf{f}$. Defining $\zeta = \frac{1}{1 - \Delta t \mu(\hat{L})}$ and $\eta = 1 + \Delta t \gamma$ and by the fact that $\|\theta_j\| = \|\hat{\theta}_j\|$, it follows

$$\begin{aligned}
\|\theta_j\| &\leq \zeta\eta \|\theta_{j-1}\| + \Delta t\, \zeta \|\hat{R}_j\| \leq (\zeta\eta)^j \|\theta_0\| + \Delta t \sum_{k=1}^{j} \zeta^k \eta^{k-1} \|\hat{R}_{j-k+1}\| \\
&\leq \Delta t \zeta \left(\sum_{k=0}^{j-1} (\zeta\eta)^{2k} \sum_{k=1}^{j} \|\hat{R}_k\|^2 \right)^{1/2},
\end{aligned}$$

where we note that $\theta_0 = 0$ and ζ is positive due to the assumption on the time step Δt. Let us define $q = \sum_{k=0}^{N_t - 1} (\zeta\eta)^{2k}$. Recalling the definition of $\hat{R}_k$

$$\hat{R}_k = V_Y^\top L \left(y^k - V_Y V_Y^\top y^k \right) + V_Y^\top \left(f_{k-1} - \mathbb{P}\mathbf{f}(V_Y V_Y^\top y^{k-1}, u^{k-1}, t^{k-1}) \right),$$

we note that

$$\begin{aligned}
&\| V_Y^\top \left(f_{k-1} - \mathbb{P}\mathbf{f}(V_Y V_Y^\top y^{k-1}, u^{k-1}, t^{k-1}) \right) \| \\
&= \| V_Y^\top \left(f_{k-1} - \mathbb{P} f_{k-1} + \mathbb{P} f_{k-1} - \mathbb{P}\mathbf{f}(V_Y V_Y^\top y^{k-1}, u^{k-1}, t^{k-1}) \right) \| \\
&\leq \mathbf{c} \| V_Y^\top \| \| f_{k-1} - V_F V_F^\top f_{k-1} \| + \gamma \|\rho_{k-1}\|
\end{aligned}$$

where we applied Proposition 1 from [29] with $\mathbf{c} = \prod_{m=1}^{d} \|(\mathbf{P_m}{}^\top \Phi_\mathbf{m})^{-1}\|$ and the Lipschitz-continuity of the function $\mathbf{f}$. Therefore, we obtain the following upper bound for the term $\hat{R}_k$

$$\|\hat{R}_k\| \leq \alpha\|\rho_k\| + \beta\|w_{k-1}\|$$

where $\alpha = \|V_Y^\top L\| + \gamma$, $\beta = \mathbf{c}\|V_Y^\top\|$ and $w_{k-1} = f_{k-1} - V_F V_F^\top f_{k-1}$.

From these results we can get the following estimate for the generic term θ_j

$$\|\theta_j\|^2 \leq (\Delta t\zeta)^2 q \sum_{k=1}^{j} \|\hat{R}_k\|^2 \leq 2(\Delta t\zeta)^2 q \sum_{k=1}^{j} (\alpha^2\|\rho_k\|^2 + \beta^2\|w_{k-1}\|^2)$$

and finally

$$\sum_{j=0}^{N_t} \|E_j\|^2 = \sum_{j=0}^{N_t} \|\rho_j\|^2 + \sum_{j=1}^{N_t} \|\theta_j\|^2 \leq C(T)\left(\mathcal{E}_y + \mathcal{E}_f\right)$$

where

$$C(T) = \max\{1 + 2q\zeta^2 T\Delta t\,\alpha^2, 2q\zeta^2 T\Delta t\beta^2\}.$$

□

Remark A.1 Supposing that $\gamma \leq -\mu(\hat{L})$, then $\zeta\eta < 1$ and we obtain the following upper bound for the quantity q

$$q = \sum_{k=0}^{N_t-1} (\zeta\eta)^{2k} \leq \frac{1}{1-(\zeta\eta)^{2N_t}}.$$

Remark A.2 The constant $C(T)$ depends on the coefficient $\mathbf{c} = \prod_{m=1}^{d} \|(\mathbf{P_m}{}^\top \Phi_\mathbf{m})^{-1}\|$, which is minimized applying the `q-deim` procedure, we refer to [20] for more details.

Acknowledgements A large part of this project was started together with Prof. Maurizio Falcone, shortly before he passed away. We greatly acknowledge his contributions, inspirations and leadership in this project. May his memory live a long time in the mathematical community.

Funding Open access funding provided by Scuola Normale Superiore within the CRUI-CARE Agreement. The second author is a member of INDAM GNCS (Gruppo Nazionale di Calcolo Scientifico). This research has been partially supported by the INdAM-GNCS project "Finanziamento Giovani Ricercatori 2020–2021".

Data Availability Matlab codes implementing the numerical examples are available at https://github.com/saluzzi/Multilinear_HJB_POD.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Akian, M., Gaubert, S., Lakhoua, A.: The max-plus finite element method for solving deterministic optimal control problems: basic properties and convergence analysis. SIAM J. Control. Optim. **47**(2), 817–848 (2008). https://doi.org/10.1137/060655286
2. Akian, M., Gaubert, S., Liu, S.: An adaptive multi-level max-plus method for deterministic optimal control problems. Preprint at arXiv:2304.10342 (2023)
3. Alla, A., Falcone, M., Saluzzi, L.: An efficient DP algorithm on a tree-structure for finite horizon optimal control problems. SIAM J. Sci. Comput. **41**(4), A2384–A2406 (2019)
4. Alla, A., Falcone, M., Saluzzi, L.: High-order approximation of the finite horizon control problem via a tree structure algorithm. IFAC-PapersOnLine **52**(2), 19–24 (2019)
5. Alla, A., Falcone, M., Saluzzi, L.: A tree structure algorithm for optimal control problems with state constraints. Rendiconti di Matematica e delle Sue Applicazioni **41**, 193–221 (2020)
6. Alla, A., Oliveira, H., Santin, G.: HJB-RBF based approach for the control of PDEs. J. Sci. Comput. **96**(1), 25 (2023)
7. Alla, A., Saluzzi, L.: A HJB-POD approach for the control of nonlinear PDEs on a tree structure. Appl. Numer. Math. **155**, 192–207 (2020). https://doi.org/10.1016/j.apnum.2019.11.023
8. Alla, A., Saluzzi, L., et al.: Feedback reconstruction techniques for optimal control problems on a tree structure. In: WCCM-ECCOMAS CONGRESS. Scipedia SL (2022)
9. Aragone, L.S., Parente, L.A., Philipp, E.A.: Fully discrete schemes for monotone optimal control problems. Comput. Appl. Math. **37**, 1047–1065 (2018)
10. Bardi, M., Capuzzo-Dolcetta, I.: Optimal Control and Viscosity Solutions of Hamilton–Jacobi–Bellman Equations. Modern Birkhäuser Classics, Birkhäuser Boston (2008)
11. Barron, E.N.: Viscosity solutions for the monotone control problem. SIAM J. Control. Optim. **23**(2), 161–171 (1985)
12. Benner, P., Bujanović, Z., Kürschner, P., Saak, J.: A numerical comparison of different solvers for large-scale, continuous-time algebraic Riccati equations and LQR problems. SIAM J. Sci. Comput. **42**(2), A957–A996 (2020). https://doi.org/10.1137/18m1220960
13. Chaturantabut, S., Sorensen, D.C.: Nonlinear model reduction via discrete empirical interpolation. SIAM J. Sci. Comput. **32**(5), 2737–2764 (2010)
14. Chaturantabut, S., Sorensen, D.C.: A state space error estimate for POD-DEIM nonlinear model reduction. SIAM J. Numer. Anal. **50**(1), 46–63 (2012)
15. Darbon, J., Dower, P.M., Meng, T.: Neural network architectures using min-plus algebra for solving certain high-dimensional optimal control problems and Hamilton–Jacobi PDEs. Math. Control Signals Syst. **35**(1), 1–44 (2023)
16. Darbon, J., Langlois, G.P., Meng, T.: Overcoming the curse of dimensionality for some Hamilton–Jacobi partial differential equations via neural network architectures. Res. Math. Sci. (2020). https://doi.org/10.1007/s40687-020-00215-6
17. D'Autilia, M.C., Sgura, I., Simoncini, V.: Matrix-oriented discretization methods for reaction–diffusion PDEs: comparisons and applications. Comput. Math. Appl. 2067–2085 (2020)
18. Dieci, L.: Numerical integration of the differential Riccati equation and some related issues. SIAM J. Numer. Anal. **29**(3), 781–815 (1992)
19. Dolgov, S., Kalise, D., Saluzzi, L.: Data-driven tensor train gradient cross approximation for Hamilton–Jacobi–Bellman equations. Preprint at arXiv:2205.05109 (2022)
20. Drmač, Z., Gugercin, S.: A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions. SIAM J. Sci. Comput. **38**(2), A631–A648 (2016)
21. Falcone, M., Ferretti, R.: Semi-Lagrangian approximation schemes for linear and Hamilton–Jacobi equations. SIAM (2013)
22. Falcone, M., Kirsten, G., Saluzzi, L.: Approximation of optimal control problems for the Navier–Stokes equation via multilinear HJB-POD. Appl. Math. Comput. **442**, 127722 (2023). https://doi.org/10.1016/j.amc.2022.127722
23. Gao, Q., Zou, M.: An analytical solution for two and three dimensional nonlinear Burgers' equation. Appl. Math. Modell. **45**, 255–270 (2017). https://doi.org/10.1016/j.apm.2016.12.018
24. Garcke, J., Kröner, A.: Suboptimal feedback control of PDEs by solving HJB equations on adaptive sparse grids. J. Sci. Comput. **70**(1), 1–28 (2016). https://doi.org/10.1007/s10915-016-0240-7
25. Golub, G.H., van Loan, C.F.: Matrix Computations, 4th edn. Johns Hopkins University Press, Baltimore (2013)
26. Han, J., Jentzen, A.E.W.: Solving high-dimensional partial differential equations using deep learning. Proc. Natl. Acad. Sci. **115**(34), 8505–8510 (2018). https://doi.org/10.1073/pnas.1718942115

27. Hinze, M., Volkwein, S.: Proper orthogonal decomposition surrogate models for nonlinear dynamical systems: error estimates and suboptimal control. In: Dimension Reduction of Large-Scale Systems, pp. 261–306. Springer (2005)
28. Hundsdorfer, W.H., Verwer, J.G., Hundsdorfer, W.: Numerical Solution of Time-Dependent Advection–Diffusion–Reaction Equations, vol. 33. Springer (2003)
29. Kirsten, G.: Multilinear POD-DEIM model reduction for 2D and 3D nonlinear systems of differential equations. J. Comput. Dyn. **9**(2), 159–183 (2022)
30. Kirsten, G., Simoncini, V.: A matrix-oriented POD-DEIM algorithm applied to nonlinear differential matrix equations (2020). Preprint at arXiv:2006.13289
31. Kirsten, G., Simoncini, V.: Order reduction methods for solving large-scale differential matrix Riccati equations. SIAM J. Sci. Comput. **42**(4), A2182–A2205 (2020)
32. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009)
33. Kunisch, K., Volkwein, S.: Optimal snapshot location for computing POD basis functions. ESAIM Math. Model. Numer. Anal. **44**(3), 509–529 (2010)
34. Kunisch, K., Volkwein, S., Xie, L.: HJB-POD based feedback design for the optimal control of evolution problems. SIAM J. Appl. Dyn. Syst. **4**, 701–722 (2004)
35. Kunisch, K., Walter, D.: Semiglobal optimal feedback stabilization of autonomous systems via deep neural network approximation. ESAIM Control Optim. Calc. Var. **27**, 16 (2021). https://doi.org/10.1051/cocv/2021009
36. Kunisch, K., Xie, L.: POD-based feedback control of burgers equation by solving the evolutionary HJB equation. Comput. Math. Appl. **49**, 1113–1126 (2005)
37. McEneaney, W.M.: A curse-of-dimensionality-free numerical method for solution of certain HJB PDEs. SIAM J. Control. Optim. **46**(4), 1239–1276 (2007). https://doi.org/10.1137/040610830
38. Meng, T., Zhang, Z., Darbon, J., Karniadakis, G.E.: SympOCnet: solving optimal control problems with applications to high-dimensional multi-agent path planning problems (2022). https://doi.org/10.48550/ARXIV.2201.05475
39. 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 (2021). https://doi.org/10.23919/ecc54610.2021.9655103
40. Oster, M., Sallandt, L., Schneider, R.: Approximating optimal feedback controllers of finite horizon control problems using hierarchical tensor formats. SIAM J. Sci. Comput. **44**(3), B746–B770 (2022)
41. Palitta, D., Simoncini, V.: Matrix-equation-based strategies for convection–diffusion equations. BIT Numer. Math. **56**(2), 751–776 (2016)
42. Philipp, E.A., Aragone, L.S., Parente, L.A.: Discrete time schemes for optimal control problems with monotone controls. Comput. Appl. Math. **34**(3), 847–863 (2015)
43. Pichi, F., Strazzullo, M., Ballarin, F., Rozza, G.: Driving bifurcating parametrized nonlinear PDEs by optimal control strategies: application to Navier–Stokes equations with model order reduction. ESAIM Math. Model. Numer. Anal. **56**(4), 1361–1400 (2022)
44. Quarteroni, A., Rozza, G.: Numerical solution of parametrized Navier–Stokes equations by reduced basis methods. Numer. Methods Part. Differ. Equ. **23**(4), 923–948 (2007)
45. Richter, L., Sallandt, L., Nüsken, N.: Solving high-dimensional parabolic PDEs using the tensor train format. In: International Conference on Machine Learning, pp. 8998–9009 (2021)
46. 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. Proc. Natl. Acad. Sci. **117**(17), 9183–9193 (2020)
47. Saluzzi, L., Alla, A., Falcone, M.: Error estimates for a tree structure algorithm solving finite horizon control problem. Preprint at arXiv:1812.11194 (2020)
48. Saluzzi, L., Alla, A., Falcone, M.: Error estimates for a tree structure algorithm solving finite horizon control problems. ESAIM Control Optim. Calc. Var **28** (2022)
49. Simoncini, V.: Computational methods for linear matrix equations. SIAM Rev. **58**(3), 377–441 (2016)
50. Simoncini, V.: Numerical solution of a class of third order tensor linear equations. Bollettino dell'Unione Matematica Italiana **13**(3), 429–439 (2020)
51. Söderlind, G.: The logarithmic norm. History and modern theory. BIT Numer. Math. **46**, 631–652 (2006)
52. Sorensen, D.C., Embree, M.: A DEIM induced CUR factorization. SIAM J. Sci. Comput. **38**(3), A1454–A1482 (2016)
53. Stabile, G., Rozza, G.: Finite volume POD-Galerkin stabilized reduced order methods for the parametrized incompressible Navier–Stokes equations. Comput. Fluids **173**, 923–948 (2018)
54. Vannieuwenhoven, N., Vandebril, R., Meerbergen, K.: A new truncation strategy for the higher-order singular value decomposition. SIAM J. Sci. Comput. **34**(2), A1027–A1052 (2012)

55. Volkwein, S.: Model reduction using proper orthogonal decomposition. In: Lecture Notes, , vol. 1025, Institute of Mathematics and Scientific Computing, University of Graz (2011)
56. Zhou, M., Han, J., Lu, J.: Actor-critic method for high dimensional static Hamilton–Jacobi–Bellman partial differential equations based on neural networks. SIAM J. Sci. Comput. **43**(6), A4043–A4066 (2021). https://doi.org/10.1137/21m1402303