ESAIM: M2AN 59 (2025) 291–330
https://doi.org/10.1051/m2an/2024074

ESAIM: Mathematical Modelling and Numerical Analysis
www.esaim-m2an.org

BE GREEDY AND LEARN: EFFICIENT AND CERTIFIED ALGORITHMS FOR PARAMETRIZED OPTIMAL CONTROL PROBLEMS

Hendrik Kleikamp[1,*], Martin Lazar[2] and Cesare Molinari[3]

Abstract. We consider parametrized linear-quadratic optimal control problems and provide their online-efficient solutions by combining greedy reduced basis methods and machine learning algorithms. To this end, we first extend the greedy control algorithm, which builds a reduced basis for the manifold of optimal final time adjoint states, to the setting where the objective functional consists of a penalty term measuring the deviation from a desired state and a term describing the control energy. Afterwards, we apply machine learning surrogates to accelerate the online evaluation of the reduced model. The error estimates proven for the greedy procedure are further transferred to the machine learning models and thus allow for efficient *a posteriori* error certification. We discuss the computational costs of all considered methods in detail and show by means of two numerical examples the tremendous potential of the proposed methodology.

Mathematics Subject Classification. 49N10, 68T07, 46E22, 62J02.

Received August 07, 2023. Accepted September 28, 2024.

1. Introduction

In this work, we are concerned with a family of parameter-dependent optimal control problems where the state equations are given as a linear, time-invariant, infinite-dimensional system. The objective functional is a quadratic functional that measures the deviation from a desired final state and an energy of the control. For a given parameter the problem consists of finding a control that minimizes the objective functional under the constraint of the state equation.

In this setting, one is typically interested in solving the optimal control problem several times for different parameters – either in a multi-query or real-time context. In either case, solving the exact optimal control problem for lots of parameters is prohibitively costly, in particular if the dimension of the state space is large. We therefore aim to develop a reduced order model that is built during a (potentially costly) offline phase by computing the exact optimal control only for few, carefully selected parameters. Afterwards, the reduced model can be explored efficiently during the online phase for previously unconsidered parameters.

Keywords and phrases. Parametrized optimal control, greedy algorithm, machine learning, deep neural networks, kernel methods, error estimation.

[1] Institute for Analysis and Numerics, Mathematics Münster, University of Münster, Einsteinstrasse 62, 48149 Münster, Germany.

[2] Department of Electrical Engineering and Computing, University of Dubrovnik, Ćira Carića 4, 20 000 Dubrovnik, Croatia.

[3] Dipartimento di Matematica, University of Genova, Via Dodecaneso 35, 16146 Genoa, Italy.

*Corresponding author: hendrik.kleikamp@uni-muenster.de

The reduced basis algorithms were successfully developed and applied to parameter-dependent control problems during the last decade (*e.g.* [2, 10, 29]), with the offline phase mainly exploring proper orthogonal decomposition (POD) or a greedy sampling procedure or their combination (POD-greedy) in case of time-dependent reduced basis methods [22]. However, the cost of the corresponding online phase might still appear high, as the computation of the projection (of the solution to a reduced basis) relies on the full-order systems. An alternative approach would employ novel numerical tools that can handle high-dimensional problems and face the curse of dimensionality.

To this end we combine model reduction and machine learning techniques. Although recently many papers appeared in which these two methods are combined, see for instance [9], the results are rather scarce when it comes to control systems [28]. In [8, 21], non-intrusive reduced basis methods that rely on neural networks were successfully applied for computing solutions to parametric partial differential equations (PDEs). We want to develop a similar approach with the aim of efficiently treating parameter-dependent control problems. The idea is to train neural networks (or some other machine learning tool) to accurately predict the coefficients of solutions in a reduced basis, with a computational effort that is independent of the dimension of the full-order model. The training is performed in the offline phase with a negligible cost, since the required data are generated by the greedy algorithm itself anyway, irrespective of a possible application of machine learning tools. As we shall see, such an approach will enable a significant computational speedup of the online phase, not only when compared to computation of the exact solutions from scratch, but also in comparison with standard reduced basis approaches, which fully justifies its development.

The paper is organized as follows: We first introduce the problem setting for parametrized linear-quadratic optimal control problems and derive the corresponding optimality system in Section 2. Afterwards, in Section 3, we describe a greedy algorithm to construct a reduced order model for parametrized optimal control problems. Additionally, we prove that the presented greedy algorithm is a weak greedy algorithm and derive *a priori* and *a posteriori* error estimates. In Section 4 we introduce a machine learning based approach to accelerate the online computation of approximate optimal controls. We further describe three different machine learning methods that are applied in our numerical experiments: deep neural networks, kernel methods and Gaussian process regression. The subsequent Section 5 discusses in detail the required computational costs for the offline and the online stages of all described algorithms. In Section 6 we show the potential of our proposed algorithms on two examples coming from the field of optimal control of PDEs. The paper concludes in Section 7 with a discussion of the theoretical and practical results and an outlook to future research related to this contribution.

2. Parametrized optimal control problems

We are interested in parametrized linear optimal control problems where the objective functional consists of a term that penalizes the deviation from a target state at final time and a term measuring the energy of the applied control. The state equation serves as a constraint and is given in form of a linear time-invariant system with linear control. The parameter enters the operators governing the state equation as well as the initial conditions and the target state. In this section, we first give a detailed definition of the problem we consider. Afterwards, the associated optimality system is introduced and a linear system of equations for the optimal final time adjoint state is derived.

2.1. Notation and problem definition

Let X and U be real Hilbert spaces with scalar products $\langle\cdot,\cdot\rangle_X$ and $\langle\cdot,\cdot\rangle_U$ as well as associated norms $\|\cdot\|_X$ and $\|\cdot\|_U$, respectively. We omit the index in the scalar product and in the corresponding norm when the spaces are clear from the context. Let $\mathcal{L}(U,X)$ denote, for instance, the set of linear and bounded operators from U to X. In the following, we will refer to X as the state and to U as the control space. We consider parametrized linear control systems of the form

$$\begin{aligned} \dot{x}_\mu(t) &= A_\mu x_\mu(t) + B_\mu u_\mu(t), \qquad t \in [0,T], \\ x_\mu(0) &= x_\mu^0, \end{aligned} \tag{1}$$

where $\mu \in \mathcal{P}$ denotes the parameter from a compact subset $\mathcal{P}$ of some Banach space (that can also be infinite-dimensional), $A_\mu \in \mathcal{L}(X,X)$ and $B_\mu \in \mathcal{L}(U,X)$ are parameter-dependent operators, $x_\mu^0 \in X$ is the parameter-dependent initial state, $x_\mu \colon [0,T] \to X$ denotes the state trajectory, $u_\mu \colon [0,T] \to U$ is the control, and $T > 0$ is the final time. Below, we consider the problem of steering the control system in (1) close to a given (potentially parameter-dependent) target state $x_\mu^T \in X$. The following (natural) assumption is supposed to hold throughout the rest of the paper:

Assumption 2.1 (Lipschitz continuity of parameter to system components maps)**.** *The subset $\mathcal{P}$ is compact and the mappings $\mathcal{P} \ni \mu \mapsto A_\mu \in \mathcal{L}(X,X)$ and $\mathcal{P} \ni \mu \mapsto B_\mu \in \mathcal{L}(U,X)$ from parameter to system matrices are Lipschitz continuous. In addition, we assume that the mappings $\mathcal{P} \ni \mu \mapsto x_\mu^0 \in X$ and $\mathcal{P} \ni \mu \mapsto x_\mu^T \in X$ from parameter to initial and target state are Lipschitz continuous.*

Furthermore, we introduce certain function spaces for the controls and states: As mentioned above, we denote by U the space of admissible controls at fixed times. The associated space of time-dependent controls is given as $G \coloneqq L^2([0,T];U)$. Similarly, we have defined the state space X. We also define the space of time-dependent states $H \coloneqq L^2([0,T];X) \cap C^1((0,T);X)$.

Given a parameter $\mu \in \mathcal{P}$, we are interested in finding a control $u_\mu^* \in G$ that minimizes the functional $\mathcal{J}_\mu \colon G \to \mathbb{R}$, defined for a control $u \in G$ as

$$\mathcal{J}_\mu(u) \coloneqq \frac{1}{2}\left[\langle x_\mu(T) - x_\mu^T, M\left(x_\mu(T) - x_\mu^T\right)\rangle + \int_0^T \langle u(t), Ru(t)\rangle \,\mathrm{d}t\right], \tag{2}$$

where $M \in \mathcal{L}(X,X)$ and $R \in \mathcal{L}(U,U)$ satisfy the following assumptions.

Assumption 2.2 (Properties of the weighting operators)**.** *The operator $M \in \mathcal{L}(X,X)$ is self-adjoint and positive-semidefinite, while $R \in \mathcal{L}(U,U)$ is self-adjoint and strictly positive-definite, meaning that $R \geq \alpha I$ for some $\alpha > 0$.*

The strict positive-definiteness of R implies in particular that R is invertible and that $\mathcal{J}_\mu$ is strongly convex with respect to the control u and thus possesses a unique minimizer. In addition, $x_\mu \in H$ is the solution of equation (1) associated to the control $u \in G$, and $x_\mu^T \in X$ is the target state. For notational simplicity, we omit stating explicitly the dependence of the state trajectory x_μ on the control u. The first term in the functional $\mathcal{J}$ penalizes a deviation of the state at the final time T from the target state x_μ^T, where the deviation in different components can be weighted by the operator M. The second term measures the energy of the control with respect to the weighting operator R. We can summarize our parametrized, linear-quadratic optimal control problem as follows:

$$\min_{u \in G} \mathcal{J}_\mu(u), \quad \text{subject to } \dot{x}_\mu(t) = A_\mu x_\mu(t) + B_\mu u(t) \text{ for } t \in [0,T], \quad x_\mu(0) = x_\mu^0. \tag{3}$$

Under the given integrability assumptions on the control, the state equation has a unique solution by Carathéodory's existence theorem, see for instance Chapter I.5. of [19]. As already stated above, the objective functional $\mathcal{J}_\mu$ is strongly convex with respect to the control u. Moreover, it is quadratic and so lower-semicontinuous. Hence, the optimal control problem in equation (3) is well-posed and has a unique solution, see for instance Corollary 2.20 in [39].

Remark 2.3 (Parameter-dependent weighting operators)**.** It is also possible to consider parameter-dependent weighting operators $M_\mu \in \mathcal{L}(X,X)$ and $R_\mu \in \mathcal{L}(U,U)$ that change for different parameters $\mu \in \mathcal{P}$. Under the assumption that the maps $\mathcal{P} \ni \mu \mapsto M_\mu \in \mathcal{L}(X,X)$ and $\mathcal{P} \ni \mu \mapsto R_\mu \in \mathcal{L}(U,U)$ are Lipschitz continuous, the theory and the algorithms developed below would not change, one solely has to replace M by M_μ and R by R_μ, respectively. For notational simplicity and since we do not consider this case in our numerical experiments,

we stick to the setting of parameter-independent operators M and R. In the case of parameter-independent self-adjoint and positive-definite weighting operators, the weighting operators can also be interpreted as the introduction of an additional scalar product on the spaces X and U, respectively. These scalar products are equivalent to the standard Euclidean scalar product if the spaces X and U are finite-dimensional.

In the subsequent section, we present and discuss the optimality system associated to the optimal control problem (3). As we will see, solving this optimality system can become costly already for a single parameter in case of moderate to large scale systems.

2.2. Linear-quadratic optimal control and the optimality system

By means of methods from the calculus of variations, one can derive the following theorem that characterizes the optimal control by considering the adjoint system.

Theorem 2.4 (Optimality system for the optimal control problem)**.** *Let $\mu \in \mathcal{P}$ be a parameter, $u_\mu^* \in G$ an optimal control, i.e. a solution of (3), and denote by $x_\mu^* \in H$ the associated state trajectory, i.e. the solution of (1) for the control u_μ^*. Then there exists an adjoint solution $\varphi_\mu^* \in H$, such that the linear boundary value problem*

$$\begin{aligned} \dot{x}_\mu(t) &= A_\mu x_\mu(t) + B_\mu u_\mu(t), \\ -\dot{\varphi}_\mu(t) &= A_\mu^* \varphi_\mu(t), \\ u_\mu(t) &= -R^{-1} B_\mu^* \varphi_\mu(t), \end{aligned} \tag{4a}$$

for $t \in [0,T]$ with initial respectively terminal conditions

$$x_\mu(0) = x_\mu^0, \qquad \varphi_\mu(T) = M\left(x_\mu(T) - x_\mu^T\right), \tag{4b}$$

is solved by $x_\mu = x_\mu^*$, $\varphi_\mu = \varphi_\mu^*$ and $u_\mu = u_\mu^*$. In equation (4a), $A_\mu^* \in \mathcal{L}(X,X)$ and $B_\mu^* \in \mathcal{L}(X,U)$ denote the adjoint operators of A_μ and B_μ, respectively.

Proof. See Appendix A. □

The optimality system in equation (4) shows that u_μ^*, x_μ^* and φ_μ^* are already uniquely determined by the optimal final time adjoint $\varphi_\mu^*(T)$. To be more precise, we can explicitly compute $u_\mu^*(t)$, $x_\mu^*(t)$ and $\varphi_\mu^*(t)$ from $\varphi_\mu^*(T)$ for $t \in [0,T]$ using the following equations, see also Figure 1:

$$\varphi_\mu^*(t) = e^{A_\mu^*(T-t)} \varphi_\mu^*(T), \tag{5}$$

$$u_\mu^*(t) = -R^{-1} B_\mu^* \varphi_\mu^*(t), \tag{6}$$

$$x_\mu^*(t) = e^{A_\mu t} x_\mu^0 - \int_0^t e^{A_\mu(t-s)} B_\mu R^{-1} B_\mu^* e^{A_\mu^*(T-s)} \varphi_\mu^*(T)\, \mathrm{d}s. \tag{7}$$

However, according to the boundary condition in equation (4b), the optimal final time adjoint $\varphi_\mu^*(T)$ is coupled to the optimal state $x_\mu^*(T)$ at the final time T, and hence, cannot be computed directly. To circumvent this issue, we compute $x_\mu^*(T)$ in terms of $\varphi_\mu^*(T)$ to obtain a linear equation for $\varphi_\mu^*(T)$. It holds

$$x_\mu^*(T) = e^{A_\mu T} x_\mu^0 - \int_0^T e^{A_\mu(T-s)} B_\mu R^{-1} B_\mu^* e^{A_\mu^*(T-s)}\, \mathrm{d}s \cdot \varphi_\mu^*(T) = e^{A_\mu T} x_\mu^0 - \Lambda_\mu^R \varphi_\mu^*(T),$$

where the *weighted controllability Gramian* $\Lambda_\mu^R \in \mathcal{L}(X,X)$ is defined as

$$\Lambda_\mu^R := \int_0^T e^{A_\mu(T-s)} B_\mu R^{-1} B_\mu^* e^{A_\mu^*(T-s)}\, \mathrm{d}s. \tag{8}$$

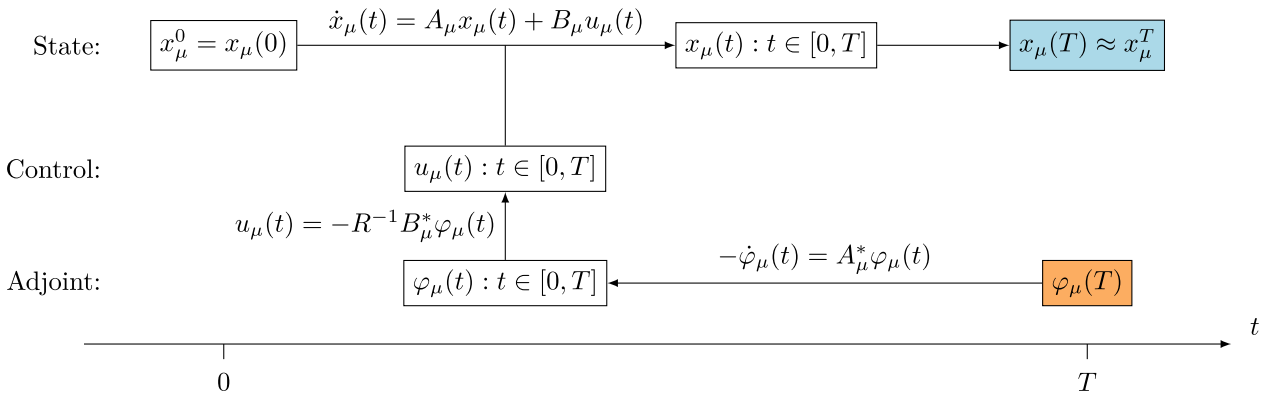

FIGURE 1. Schematic visualization of the computations required to approach the final time state $x_\mu(T)$ (marked in blue) from a final time adjoint $\varphi_\mu(T)$ (marked in orange).

The Gramian Λ_μ^R is bounded due to the boundedness of all involved operators and the finite time horizon we consider. According to Theorem 2.4, we have that

$$\varphi_\mu^*(T) = M\left(x_\mu^*(T) - x_\mu^T\right),$$

and it further holds

$$x_\mu^*(T) = e^{A_\mu T} x_\mu^0 - \Lambda_\mu^R \varphi_\mu^*(T) \tag{9}$$

as shown above. Combining these two equations, we obtain

$$\varphi_\mu^*(T) = M\left(e^{A_\mu T} x_\mu^0 - x_\mu^T - \Lambda_\mu^R \varphi_\mu^*(T)\right).$$

Rearranging the previous equation gives the following lemma that states the resulting equation the optimal final time adjoint $\varphi_\mu^*(T) \in X$ has to solve:

Lemma 2.5 (Linear system for the optimal final time adjoint)**.** *Let $\varphi_\mu^*(T)$ denote the optimal adjoint state at time T that determines the solution of the optimality system (4). Then it holds*

$$\left(I + M\Lambda_\mu^R\right)\varphi_\mu^*(T) = M\left(e^{A_\mu T} x_\mu^0 - x_\mu^T\right), \tag{10}$$

where $I \in \mathcal{L}(X,X)$ denotes the identity.

Since the operator R is positive-definite, the same holds for R^{-1}. Therefore, the Gramian Λ_μ^R is self-adjoint and positive-semidefinite. We also emphasize at this point that the controllability Gramian as defined here depends on the parameter $\mu \in \mathcal{P}$ in the same way as the involved system operators do.

We make the following assumption that is supposed to hold throughout the rest of the paper and will become useful for the error estimation in Section 3:

Assumption 2.6 (Positivity of the product of the weighting and the Gramian operator)**.** *We assume that the operator $M\Lambda_\mu^R \in \mathcal{L}(X,X)$ is positive-semidefinite for all parameters $\mu \in \mathcal{P}$.*

As a simple example of a weighting operator $M \in \mathcal{L}(X,X)$ that fulfills Assumption 2.6, we will use in our numerical experiments in Section 6 the choice $M = \kappa I$ for a suitable constant $\kappa > 0$, where $I \in \mathcal{L}(X,X)$ denotes the identity. More generally, the assumption will be satisfied if M and the Gramian Λ_μ^R commute.

Remark 2.7 (Computation of products with the Gramian)**.** We emphasize that for a given vector $p \in X$, the product $\Lambda_\mu^R p$ can be computed without constructing Λ_μ^R explicitly. Instead, one uses that $-\Lambda_\mu^R p = x_\mu(T)$ (see Eq. (9)), where x_μ solves system (4) with boundary conditions $x_\mu(0) = 0$ and $\varphi_\mu(T) = p$, respectively. It is therefore sufficient to solve (4) by first solving the equation for φ_μ backward in time, then computing u_μ, and finally solving the state equation for x_μ forward in time. Assembling $\Lambda_\mu^R \in \mathcal{L}(X, X)$ (which depends on the parameter $\mu \in \mathcal{P}$) would be prohibitively costly and is infeasible for moderate to large scale systems when having to do so for multiple parameters.

Because of large computational costs required for solving the optimal control problem for a single parameter (see also the discussions in Sect. 5), we aim for building a reduced order model that replaces the equation in (10) by a linear system of small dimension that can be solved faster while still providing a sufficiently accurate approximation of the optimal solution.

3. Reduced order modeling by a greedy algorithm

In the previous section, we derived a linear system for the optimal final time adjoint state $\varphi_\mu^*(T)$ and observed that the optimal control is already completely determined by $\varphi_\mu^*(T)$. The main goal of this section is to extend the greedy control algorithm introduced in [29] to the setting of parametrized optimal control problems of the form (3).

3.1. Error estimation by considering the residual

Given a parameter $\mu \in \mathcal{P}$ and an arbitrary vector $p \in X$, we define the error estimator $\eta_\mu \colon X \to \mathbb{R}$ as

$$\eta_\mu(p) \coloneqq \left\| M\left(e^{A_\mu T} x_\mu^0 - x_\mu^T\right) - (I + M\Lambda_\mu^R)p \right\|, \tag{11}$$

where the term inside the norm corresponds to the residual of equation (10).

We obtain the following theorem which states that η_μ is an efficient and reliable error estimator for the final time adjoint state, where we use in the proof that the operator $M\Lambda_\mu^R$ is positive (see Assumption 2.6).

Theorem 3.1 (Efficient and reliable error estimator for the distance from the adjoint)**.** *Let $\mu \in \mathcal{P}$ be a parameter, $\varphi_\mu^*(T) \in X$ the corresponding optimal final time adjoint solving equation (10), and let $p \in X$ be an approximate final time adjoint. Then it holds*

$$\left\|\varphi_\mu^*(T) - p\right\| \le \eta_\mu(p) \le \left\|I + M\Lambda_\mu^R\right\|_{\mathcal{L}(X,X)} \left\|\varphi_\mu^*(T) - p\right\|. \tag{12}$$

Proof. For the optimal final time adjoint $\varphi_\mu^*(T) \in X$, according to Theorem 2.5, it holds

$$M(e^{A_\mu T} x_\mu^0 - x_\mu^T) = (I + M\Lambda_\mu^R)\varphi_\mu^*(T).$$

Plugging this equation into the definition of the error estimator, we obtain

$$\begin{aligned}
\eta_\mu(p)^2 &= \left\|(I + M\Lambda_\mu^R)(\varphi_\mu^*(T) - p)\right\|^2 \\
&= \left\|\varphi_\mu^*(T) - p)\right\|^2 + \left\|M\Lambda_\mu^R(\varphi_\mu^*(T) - p)\right\|^2 + 2\langle M\Lambda_\mu^R(\varphi_\mu^*(T) - p), (\varphi_\mu^*(T) - p)\rangle \\
&\ge \left\|\varphi_\mu^*(T) - p)\right\|^2,
\end{aligned}$$

where we used that $M\Lambda_\mu^R$ is positive. From here the lower bound in (12) follows directly. The upper bound comes simply from the definition of the operator norm. □

The error estimator $\eta_\mu(p)$ can be used to quantify the quality of an approximation of the optimal final time adjoint $\varphi_\mu^*(T)$ by an arbitrary $p \in X$ without ever computing $\varphi_\mu^*(T)$. Instead, it is only necessary to solve the linear initial value problem (4) twice: first with $x_\mu(0) = 0$, $\varphi_\mu(T) = p$ (in order to compute $-\Lambda_\mu^R p$) and then with $x_\mu(0) = x_\mu^0$, $\varphi_\mu(T) = 0$ (*i.e.* without control of the state Eq. (1)) in order to determine $e^{A_\mu T} x_\mu^0$ (the term $e^{A_\mu T} x_\mu^0$ corresponds to the free dynamics). The error estimator η_μ can therefore be evaluated much more efficiently than solving the full optimal control problem. We also remark that Theorem 3.1 implies that it holds $\eta_\mu(p) = 0$ if and only if $p = \varphi_\mu^*(T)$.

In the following section, we describe an algorithm that constructs a low-dimensional subspace of X in which an approximation of the optimal final time adjoint is searched. The basis of this subspace is chosen by a greedy algorithm that uses the error estimator η_μ to determine the next parameter and in particular the associated final time adjoint that is added to the basis.

3.2. Greedy procedure for constructing a reduced basis

Let $\varepsilon > 0$ be a prescribed tolerance. We aim at constructing a reduced subspace $X^N \subset X$ of low dimension $\dim X^N = N$, such that X^N approximates the manifold

$$\mathcal{M} \coloneqq \{\varphi_\mu^*(T) : \mu \in \mathcal{P}\} \subset X \tag{13}$$

of all possible optimal final time adjoints up to a tolerance of ε. Since $\mathcal{P}$ is compact and the mapping $\mathcal{P} \ni \mu \mapsto \varphi_\mu^*(T) \in X$ is Lipschitz continuous (which follows from Assumption 2.1 and the characterization of $\varphi_\mu^*(T)$ as solution of the linear equation in (10)), the set $\mathcal{M}$ is also compact. The distance $\operatorname{dist}(Y, Z)$ of a subspace $Y \subset X$ to a compact set $Z \subset X$ is defined as

$$\operatorname{dist}(Y, Z) \coloneqq \sup_{z \in Z} \inf_{y \in Y} \|y - z\| .$$

We thus search for a reduced space $X^N \subset X$ such that $\operatorname{dist}(X^N, \mathcal{M}) \leq \varepsilon$. To this end, greedy algorithms construct a sequence of reduced bases $\Phi^0, \Phi^1, \ldots$, starting with $\Phi^0 = \emptyset \subset X$, and corresponding reduced spaces $X^0 = \operatorname{span}(\Phi^0) = \{0\}, X^1 = \operatorname{span}(\Phi^1), \ldots$ by successively adding new basis functions. These functions are selected as snapshots, *i.e.* optimal final time adjoints for certain parameters. Assume a reduced basis $\Phi^k = \{\varphi_1, \ldots, \varphi_k\} \subset X$ of dimension k is given with associated reduced space $X^k = \operatorname{span}(\Phi^k)$. The next basis function $\varphi_{k+1} \in X$ is chosen as $\varphi_{k+1} = \varphi_{\mu_{k+1}}^*(T)$, where the parameter $\mu_{k+1} \in \mathcal{P}$ is determined such that the distance $\operatorname{dist}(X^k, \{\varphi_{\mu_{k+1}}^*(T)\})$ of $\varphi_{\mu_{k+1}}^*(T)$ to X^k (or a suitable estimate of this distance that is cheaper to compute) is the largest among all parameters $\mu \in \mathcal{P}$, *i.e.* it holds

$$\mu_{k+1} \in \operatorname{argmax}_{\mu \in \mathcal{P}} \operatorname{dist}(X^k, \{\varphi_\mu^*(T)\}).$$

After choosing the parameter μ_{k+1}, the optimal final time adjoint $\varphi_{\mu_{k+1}}^*(T)$ is computed by solving equation (10) and added to the reduced basis, *i.e.* the new basis Φ^{k+1} is defined as $\Phi^{k+1} = \Phi^k \cup \{\varphi_{k+1}\}$ for $\varphi_{k+1} = \varphi_{\mu_{k+1}}^*(T)$. After updating the reduced basis, the error for the parameter μ_{k+1} vanishes, since the optimal final time adjoint for μ_{k+1} is now contained in the reduced space X^{k+1}, *i.e.* $\varphi_{\mu_{k+1}}^*(T) \in \Phi^{k+1} \subset X^{k+1}$.

To make the construction of Φ^N computationally feasible one usually restricts the search for a new parameter to a finite set $\mathcal{P}_{\text{train}} \subset \mathcal{P}$ of $n_{\text{train}} \coloneqq |\mathcal{P}_{\text{train}}| < \infty$ training parameters. This set $\mathcal{P}_{\text{train}}$ of training parameters is chosen to be dense in the overall set $\mathcal{P}$ of parameters (in the sense that the union of balls with certain radius $\delta > 0$ specified below around the training parameters covers the whole (compact) parameter set $\mathcal{P}$, *i.e.* it holds $\mathcal{P} \subseteq \bigcup_{\mu \in \mathcal{P}_{\text{train}}} B_\delta(\mu)$, where $B_\delta(\mu) \coloneqq \{\bar{\mu} : \|\mu - \bar{\mu}\| \leq \delta\}$), see also the pseudocode of the procedure provided in Algorithm 1 below. Furthermore, instead of computing the true distance $\operatorname{dist}(X^k, \{\varphi_\mu^*(T)\})$ of a subspace X^k to $\varphi_\mu^*(T)$ for all training parameters $\mu \in \mathcal{P}_{\text{train}}$, an estimate of this distance is used that does not require the computation of $\varphi_\mu^*(T)$. In our setting, we will estimate the distance using the error estimator from equation (11) and a suitably chosen approximate final time adjoint.

Once having constructed a reduced basis Φ^k, it is used to obtain an approximate final time adjoint for an arbitrary parameter $\mu \in \mathcal{P}$. To this end, we first compute $x_i^\mu = (I + M\Lambda_\mu^R)\varphi_i \in X$ for all $i = 1, \dots, k$. The state $-\Lambda_\mu^R \varphi_i \in X$ for $i \in \{1, \dots, k\}$ corresponds to the final time state $x_\mu(T)$ of the system (4a) with zero initial datum and the control determined by the basis function $\varphi_i \in \Phi^k$. Thus, x_i^μ can be seen as the perturbation of the final state $-\Lambda_\mu^R \varphi_i \in X$ by the operator $-\left((\Lambda_\mu^R)^{-1} + M\right)$, *i.e.* it holds $-\left((\Lambda_\mu^R)^{-1} + M\right)\left(-\Lambda_\mu^R \varphi_i\right) = (I + M\Lambda_\mu^R)\varphi_i = x_i^\mu$. Hence, the space $Y_\mu^k \coloneqq \operatorname{span}(x_1^\mu, \dots, x_k^\mu) = (I + M\Lambda_\mu^R)X^k \subset X$ contains all possible perturbed final time states that can be generated from the final time adjoints in the reduced basis Φ^k for the system determined by the parameter μ and starting from zero initial conditions. This motivates, together with the linear equation for the optimal final time adjoint from Theorem 2.5, to consider the projection of $M\left(e^{A_\mu T} x_\mu^0 - x_\mu^T\right)$ onto Y_μ^k and therefore the distance $\operatorname{dist}(Y_\mu^k, \{M\left(e^{A_\mu T} x_\mu^0 - x_\mu^T\right)\})$, see also Figure 2. Let us denote by $P_Y(x) \in Y$ the orthogonal projection of a vector $x \in X$ onto a subspace $Y \subset X$. Then it holds

$$\operatorname{dist}\left(Y_\mu^k, \left\{M\left(e^{A_\mu T} x_\mu^0 - x_\mu^T\right)\right\}\right) = \left\| M\left(e^{A_\mu T} x_\mu^0 - x_\mu^T\right) - P_{Y_\mu^k}\left(M\left(e^{A_\mu T} x_\mu^0 - x_\mu^T\right)\right)\right\|.$$

We can express the projection in the basis of Y_μ^k given by the vectors $x_1^\mu, \dots, x_k^\mu$ (which form a linearly independent set since the corresponding parameters $\mu_1, \dots, \mu_k \in \mathcal{P}_{\text{train}}$ were selected by the greedy procedure). Let us write

$$P_{Y_\mu^k}\left(M\left(e^{A_\mu T} x_\mu^0 - x_\mu^T\right)\right) = \sum_{i=1}^k \alpha_i^\mu x_i^\mu \in Y_\mu^k \tag{14}$$

for some coefficients $\alpha_1^\mu, \dots, \alpha_k^\mu \in \mathbb{R}$. Then we choose the approximate final time adjoint $\tilde{\varphi}_\mu^k \in X^k$ as

$$\tilde{\varphi}_\mu^k = \sum_{i=1}^k \alpha_i^\mu \varphi_i. \tag{15}$$

With these definitions, we have that

$$(I + M\Lambda_\mu^R)\tilde{\varphi}_\mu^k = \sum_{i=1}^k \alpha_i^\mu x_i^\mu = P_{Y_\mu^k}\left(M\left(e^{A_\mu T} x_\mu^0 - x_\mu^T\right)\right),$$

and in particular

$$\eta_\mu(\tilde{\varphi}_\mu^k) = \operatorname{dist}\left(Y_\mu^k, \left\{M\left(e^{A_\mu T} x_\mu^0 - x_\mu^T\right)\right\}\right).$$

This allows us to estimate the efficiency of the reduced space X^k through the error estimator η_μ, *i.e.*

$$\operatorname{dist}\left(X^k, \left\{\varphi_\mu^*(T)\right\}\right) \approx \eta_\mu(\tilde{\varphi}_\mu^k).$$

More precisely, the following theorem holds.

Theorem 3.2 (Efficient and reliable error estimator for a reduced space)**.** *Let $\mu \in \mathcal{P}$ be a parameter and $\varphi_\mu^*(T) \in X$ the optimal final time adjoint solving the optimality system in equation (4). Then, for the error estimator $\eta_\mu(\tilde{\varphi}_\mu^k)$ with η_μ introduced in equation (11) and $\tilde{\varphi}_\mu^k$ from equation (15), it holds*

$$\operatorname{dist}(X^k, \{\varphi_\mu^*(T)\}) \le \eta_\mu(\tilde{\varphi}_\mu^k) \le \left\|I + M\Lambda_\mu^R\right\|_{\mathcal{L}(X,X)} \cdot \operatorname{dist}(X^k, \{\varphi_\mu^*(T)\}). \tag{16}$$

Proof. Due to orthogonality of the projection and the Pythagorean theorem, it holds

$$\left\|\varphi_\mu^*(T) - \tilde{\varphi}_\mu^k\right\|^2 = \left\|\varphi_\mu^*(T) - P_{X^k}(\varphi_\mu^*(T))\right\|^2 + \left\|\tilde{\varphi}_\mu^k - P_{X^k}(\varphi_\mu^*(T))\right\|^2$$

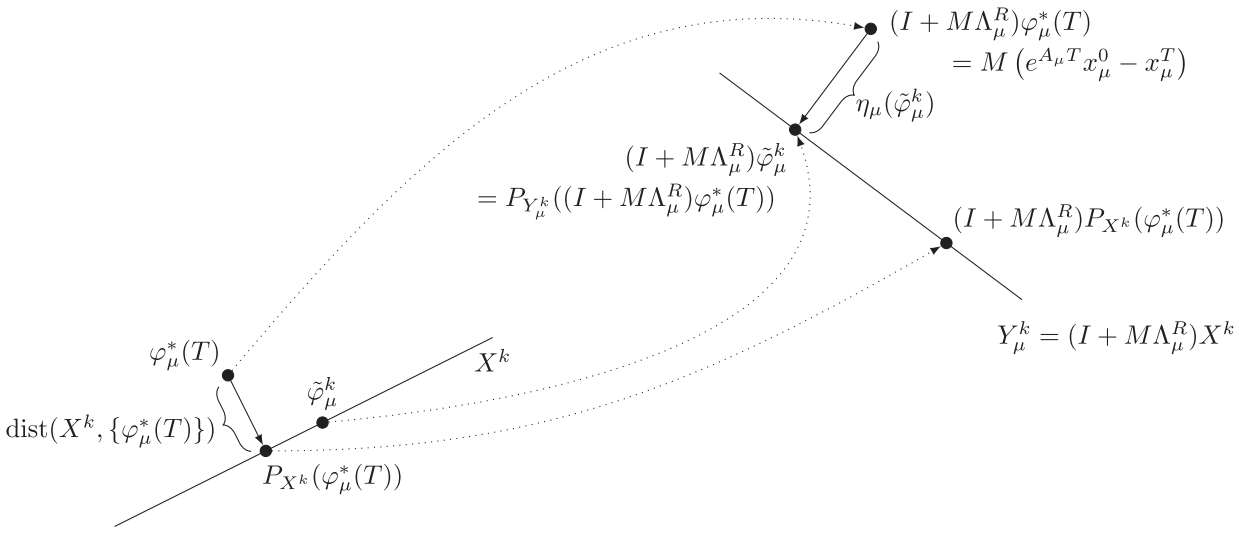

FIGURE 2. Visualization of the different final time adjoints and states that occur in the proof of Theorem 3.2. Solid arrows represent orthogonal projections onto the respective spaces X^k and Y_μ^k while dotted arrows visualize applications of $I + M\Lambda_\mu^R$.

and therefore

$$\begin{aligned}\mathrm{dist}(X^k, \{\varphi_\mu^*(T)\})^2 &= \left\|\varphi_\mu^*(T) - P_{X^k}(\varphi_\mu^*(T))\right\|^2 \\ &= \left\|\varphi_\mu^*(T) - \tilde{\varphi}_\mu^k\right\|^2 - \left\|\tilde{\varphi}_\mu^k - P_{X^k}(\varphi_\mu^*(T))\right\|^2 \\ &\leq \eta_\mu(\tilde{\varphi}_\mu^k)^2\end{aligned}$$

by Theorem 3.1. This implies the lower bound in equation (16).

In order to obtain the upper one, we estimate

$$\begin{aligned}\eta_\mu(\tilde{\varphi}_\mu^k) &= \left\|M\left(e^{A_\mu T}x_\mu^0 - x_\mu^T\right) - (I + M\Lambda_\mu^R)\tilde{\varphi}_\mu^k\right\| \\ &= \left\|(I + M\Lambda_\mu^R)\varphi_\mu^*(T) - P_{Y_\mu^k}\left((I + M\Lambda_\mu^R)\varphi_\mu^*(T)\right)\right\| \\ &\leq \left\|(I + M\Lambda_\mu^R)\varphi_\mu^*(T) - (I + M\Lambda_\mu^R)P_{X^k}(\varphi_\mu^*(T))\right\| \\ &\leq \left\|I + M\Lambda_\mu^R\right\|_{\mathcal{L}(X,X)}\left\|\varphi_\mu^*(T) - P_{X^k}(\varphi_\mu^*(T))\right\| \\ &= \left\|I + M\Lambda_\mu^R\right\|_{\mathcal{L}(X,X)}\mathrm{dist}(X^k, \{\varphi_\mu^*(T)\}).\end{aligned}$$

Here we used that $(I + M\Lambda_\mu^R)P_{X^k}(\varphi_\mu^*(T)) \in Y_\mu^k$, which follows from the fact that $P_{X^k}(\varphi_\mu^*(T)) \in X^k$ and $Y_\mu^k = (I + M\Lambda_\mu^R)X^k$. □

For a visualization of the quantities involved in the proof of Theorem 3.2 and their relationships, see Figure 2.

Remark 3.3 (On the efficiency bound in Thm. 3.2)**.** We emphasize at this point that the first bound in Theorem 3.2 holds for an arbitrary choice $p \in X^k$ for the approximate final time adjoint, *i.e.* we have $\mathrm{dist}(X^k, \{\varphi_\mu^*(T)\}) \leq \eta_\mu(p)$ for all $p \in X^k$. However, the efficiency of the error estimator $\eta_\mu(\tilde{\varphi}_\mu^k)$, *i.e.* the second inequality in Theorem 3.2, heavily relies on the choice of the approximate final time adjoint $\tilde{\varphi}_\mu^k$. The efficiency bound is required to maintain the convergence of the weak greedy algorithm, see Section 3.3.

To formulate the greedy algorithm in pseudo-code, we observe that the following properties hold, which will allow for an error analysis of the greedy algorithm in Section 3.3: The parameter to solution map, given as

$$\mathcal{P} \ni \mu \mapsto \varphi_\mu^*(T) = \left(I + M\Lambda_\mu^R\right)^{-1} M \left(e^{A_\mu T} x_\mu^0 - x_\mu^T\right) \in X,$$

is Lipschitz continuous (due to the Lipschitz continuity in Assumption 2.1, and since the function that maps an operator to its inverse is also Lipschitz continuous), as already mentioned above. The constant $C_{\varphi^*} > 0$ is defined as the Lipschitz constant of this mapping, *i.e.* it holds

$$\left\|\varphi_\mu^*(T) - \varphi_{\tilde{\mu}}^*(T)\right\| \leq C_{\varphi^*} \left\|\mu - \tilde{\mu}\right\| \qquad \text{for all } \mu, \tilde{\mu} \in \mathcal{P}.$$

Moreover, the norm $\left\|I + M\Lambda_\mu^R\right\|$ is uniformly bounded over the parameter domain. The constant $C_\Lambda > 0$ is defined as

$$C_\Lambda \coloneqq \sup_{\mu \in \mathcal{P}} \left\|I + M\Lambda_\mu^R\right\|_{\mathcal{L}(X,X)} < \infty.$$

Both of these statements are direct consequences of Assumption 2.1 and the compactness of the parameter set $\mathcal{P}$. We further remark that it holds $C_\Lambda \geq 1$, which immediately follows from equation (12), and that $C_{\varphi^*} \geq 0$. In addition to the constants above, we define $0 < \gamma \leq 1$ as

$$\gamma \coloneqq \frac{1}{C_{\varphi^*} + C_\Lambda}. \tag{17}$$

The constant γ will play a crucial role in the error analysis and in the performance of the greedy algorithm.

The procedure of the greedy algorithm is summarized in Algorithm 1.

Algorithm 1. Offline phase of the greedy procedure.

Input: Tolerance $\varepsilon > 0$

Output: Reduced basis $\Phi^N \subset X$, reduced space $X^N = \operatorname{span}\left(\Phi^N\right) \subset X$, training parameters and associated reduced coefficients collected in a set $D_{\text{train}} \subset \mathcal{P}_{\text{train}} \times \mathbb{R}^N$

procedure GREEDY(ε)
- $\tilde{\varepsilon} \leftarrow C_\Lambda \gamma \varepsilon, \quad \delta \leftarrow \frac{\tilde{\varepsilon}}{C_\Lambda}$
- define a training set $\mathcal{P}_{\text{train}} \subset \mathcal{P}$ such that for all $\mu \in \mathcal{P}$ there exists $\tilde{\mu} \in \mathcal{P}_{\text{train}}$ with $\|\mu - \tilde{\mu}\| \leq \delta$
- $k \leftarrow 0, \quad \Phi^0 \leftarrow \emptyset, \quad X^0 \leftarrow \{0\} \subset X$
- $\tilde{\varphi}_\mu^0 \leftarrow 0$ for all $\mu \in \mathcal{P}_{\text{train}}$
- select next parameter $\mu_1 \leftarrow \operatorname{argmax}_{\mu \in \mathcal{P}_{\text{train}}} \eta_\mu(\tilde{\varphi}_\mu^0)$
- **while** $\eta_{\mu_{k+1}}(\tilde{\varphi}_{\mu_{k+1}}^k) > \tilde{\varepsilon}$ **do**
  - compute optimal final time adjoint $\varphi_{k+1} \leftarrow \varphi_{\mu_{k+1}}^*(T)$ by solving eq. (10)
  - $\Phi^{k+1} \leftarrow \Phi^k \cup \{\varphi_{k+1}\}$ (apply orthonormalization using Gram-Schmidt algorithm if desired)
  - $X^{k+1} \leftarrow \operatorname{span}\left(\Phi^{k+1}\right)$
  - $k \leftarrow k + 1$
  - **for** $\mu \in \mathcal{P}_{\text{train}}$ **do**
    - compute $x_i^\mu \leftarrow (I + M\Lambda_\mu^R)\varphi_i$ for $i = 1, \ldots, k$
    - assemble operator $\bar{X}_\mu \leftarrow [x_1^\mu \ \cdots \ x_k^\mu] \in \mathcal{L}(\mathbb{R}^k, X)$ for projection onto Y_μ^k
    - compute coefficients $\alpha^\mu = (\alpha_1^\mu, \ldots, \alpha_k^\mu)^\top \in \mathbb{R}^k$ as solution of the $k \times k$ linear system (see eq. (14))
    $$\bar{X}_\mu^* \bar{X}_\mu \alpha^\mu = \bar{X}_\mu^* M \left(e^{A_\mu T} x_\mu^0 - x_\mu^T\right)$$
    - compute final time adjoint $\tilde{\varphi}_\mu^k \leftarrow \sum_{i=1}^k \alpha_i^\mu \varphi_i$
  - select next parameter $\mu_{k+1} \leftarrow \operatorname{argmax}_{\mu \in \mathcal{P}_{\text{train}}} \eta_\mu(\tilde{\varphi}_\mu^k)$
- **return** $\Phi^N \leftarrow \Phi^k$, $X^N \leftarrow X^k$, $D_{\text{train}} \leftarrow \{(\mu, \alpha^\mu) : \mu \in \mathcal{P}_{\text{train}}\}$

In the greedy procedure in Algorithm 1 we also collect and return all training parameters and the corresponding reduced coefficients in the set D_{train}. This will be convenient and useful for the machine learning training in Section 4, see in particular Algorithm 4.

Remark 3.4 (Orthonormalization of the reduced basis)**.** To improve numerical stability and to simplify computations of projections, new basis functions can be orthonormalized against the previous ones before adding them to the reduced basis. For that purpose, for instance the Gram-Schmidt algorithm can be used which results in an orthonormal reduced basis Φ^N.

Remark 3.5 (Construction of a reduced basis without using the greedy procedure)**.** Instead of applying the greedy algorithm described above, it is also possible to determine a reduced basis using methods such as the POD method [14]. This algorithm requires a set of given training snapshots $\varphi^*_{\mu_1}(T), \dots, \varphi^*_{\mu_{n_{\text{train}}}}(T) \in X$ for several training parameters $\mu_1, \dots, \mu_{n_{\text{train}}} \in \mathcal{P}$ that has to be computed beforehand. However, solving the optimal control problem many times is very costly. Since an efficient error estimator is available in our setting, it might therefore be beneficial to select the reduced basis functions in a greedy manner. Nevertheless, if enough training data is already available, the proper orthogonal decomposition might be a plausible alternative to the greedy procedure.

3.3. Error analysis of the greedy algorithm

In this section, we will investigate the error of the greedy algorithm and prove that Algorithm 1 indeed provides a reduced basis Φ^N and corresponding reduced space $X^N \subset X$ such that each optimal final time adjoint $\varphi^*_\mu(T) \in X$ can be approximated by an error of at most ε in the reduced space X^N (see Thm. 3.7). Before doing so, we give a short general introduction to (weak) greedy algorithms and recall a result by DeVore *et al.* [11] (see Thm. 3.6) stating that weak greedy algorithms produce approximating sequences of reduced spaces with the same convergence rates as the optimal spaces in the sense of Kolmogorov.

Let V be a Hilbert space and $K \subset V$ a compact subset of V. Greedy algorithms aim at constructing a sequence of subspaces $V^N \subset V$ of dimension $N \in \mathbb{N}$ that approximate the set K. Similarly to Algorithm 1 described in Section 3.2, the spaces V^N are constructed by selecting new basis functions in a "greedy" fashion, *i.e.* by selecting elements from K that are currently poorly approximated by elements from V^N. The general procedure is stated in the following pseudocode:

Algorithm 2. General (weak) greedy algorithm.

Input: Compact set $K \subset V$, greedy constant $\gamma \in (0, 1]$, tolerance $\varepsilon > 0$
Output: Reduced basis $\Psi^N \subset V$, reduced space $V^N = \operatorname{span}\left(\Psi^N\right) \subset V$

$N \leftarrow 0, \quad \Psi^N \leftarrow \emptyset, \quad V^0 \leftarrow \{0\} \subset V$
while $\operatorname{dist}(V^N, K) > \varepsilon$ **do**
  choose next element $v_{N+1} \in K$ such that it holds

$$\operatorname{dist}(V^N, \{v_{N+1}\}) \geq \gamma \cdot \operatorname{dist}(V^N, K) \tag{18}$$

  $\Psi^{N+1} \leftarrow \Psi^N \cup \{v_{N+1}\}$
  $V^{N+1} \leftarrow \operatorname{span}\left(\Psi^{N+1}\right)$
  $N \leftarrow N + 1$
return Ψ^N, V^N

For a constant $\gamma < 1$ the algorithm above is called *weak* greedy algorithm, while for $\gamma = 1$ the procedure is also known as *strong* greedy algorithm. By considering the optimal reduced subspace of dimension $N \in \mathbb{N}$, we

can measure the quality of the reduced space obtained by the weak greedy algorithm. We denote by

$$d_N(K) \coloneqq \inf_{\substack{W \subset V, \\ \dim W = N}} \operatorname{dist}(W, K) \tag{19}$$

the so-called *Kolmogorov N-width*, where W denotes an N-dimensional linear subspace of V. In this way, $d_N(K)$ measures the performance of the optimal approximation of K by a linear space of fixed dimension N.

In contrast, by $\sigma_N(K)$ we denote the error of the reduced space $V^N \subset V$ constructed by the greedy algorithm and defined as

$$\sigma_N(K) \coloneqq \operatorname{dist}(V^N, K). \tag{20}$$

The importance and efficiency of (weak) greedy algorithms is ensured by the following theorem that gives a connection between the decay of the Kolmogorov N-width $d_N(K)$ with respect to N and the decay of the error of the reduced space $\sigma_N(K)$.

Theorem 3.6 ([11], Cor. 3.3 (ii) and (iii))**.** *For the weak greedy algorithm with constant γ in a Hilbert space V and a compact set $K \subset V$, we have the following:*

(i) *If $d_N(K) \leq C_0 N^{-\alpha}$, $N = 1, 2, \ldots$, then $\sigma_N(K) \leq C_1 N^{-\alpha}$, $N = 1, 2, \ldots$, with $C_1 \coloneqq 2^{5\alpha+1}\gamma^{-2}C_0$.*
(ii) *If $d_N(K) \leq C_0 e^{-c_0 N^{\alpha}}$, $N = 1, 2, \ldots$, then $\sigma_N(K) \leq \sqrt{2C_0}\gamma^{-1}e^{-c_1 N^{\alpha}}$, $N = 1, 2, \ldots$, where $c_1 \coloneqq 2^{-1-2\alpha}c_0$.*

This theorem implies that weak greedy algorithms preserve the decay rates of the Kolmogorov N-widths, in such a way providing the optimal approximation errors, up to a multiplicative constant.

In our setting, we consider the Hilbert space $V = X$ and aim at approximating the set $K = \mathcal{M}$ of optimal final time adjoint states over the parameter domain $\mathcal{P}$ as defined in equation (13). We will prove in the following that Algorithm 1 is indeed a weak greedy algorithm as introduced in Algorithm 2, in particular that the resulting reduced spaces X^N satisfy relation (18). For this reason, according to Theorem 3.6, we can transfer convergence properties of the Kolmogorov N-width of $\mathcal{M}$ to the error decay of the spaces X^N constructed by Algorithm 1.

Theorem 3.7 (Weak greedy algorithm and approximation error)**.** *The procedure presented in Algorithm 1 is a weak greedy algorithm with the constant γ defined in equation (17). Furthermore, for the resulting reduced space $X^N \subset X$ it holds for all parameters $\mu \in \mathcal{P}$ the approximation error estimate*

$$\operatorname{dist}(X^N, \{\varphi_\mu^*(T)\}) \leq \varepsilon. \tag{21}$$

Proof. Let $k \in \{1, \ldots, N-1\}$ be an arbitrary iteration of Algorithm 1 such that the termination criterion is not fulfilled. Let further be the corresponding reduced basis Φ^k and reduced space $X^k = \operatorname{span}\left(\Phi^k\right)$ given. We have to prove that it holds

$$\operatorname{dist}(X^k, \{\varphi_{\mu_{k+1}}^*(T)\}) \geq \gamma \cdot \max_{\mu \in \mathcal{P}} \operatorname{dist}(X^k, \{\varphi_\mu^*(T)\}) = \gamma \cdot \operatorname{dist}(X^k, \mathcal{M}), \tag{22}$$

with a constant γ independent of k (see also Eq. (18)). To this end, let $\mu \in \mathcal{P}$ be an arbitrary parameter. Then there exists a parameter $\tilde{\mu} \in \mathcal{P}_{\text{train}}$, such that $\|\mu - \tilde{\mu}\| \leq \delta$. Due to the choice of the parameter $\mu_{k+1} \in \mathcal{P}_{\text{train}}$, it holds

$$\eta_{\mu_{k+1}}(\tilde{\varphi}_{\mu_{k+1}}^k) \geq \eta_{\tilde{\mu}}(\tilde{\varphi}_{\tilde{\mu}}^k)$$

for all $\tilde{\mu} \in \mathcal{P}_{\text{train}}$. We then have the estimate

$$\begin{aligned}
\operatorname{dist}(X^k, \{\varphi_\mu^*(T)\}) &\leq \left\|\varphi_\mu^*(T) - \varphi_{\tilde{\mu}}^*(T)\right\| + \operatorname{dist}(X^k, \{\varphi_{\tilde{\mu}}^*(T)\}) \\
&\leq C_{\varphi^*}\delta + \operatorname{dist}(X^k, \{\varphi_{\tilde{\mu}}^*(T)\}) \\
&\leq C_{\varphi^*}\frac{\tilde{\varepsilon}}{C_\Lambda} + \eta_{\tilde{\mu}}(\tilde{\varphi}_{\tilde{\mu}}^k) \\
&\leq C_{\varphi^*}\frac{\tilde{\varepsilon}}{C_\Lambda} + \eta_{\mu_{k+1}}(\tilde{\varphi}_{\mu_{k+1}}^k) \\
&\leq \left(\frac{C_{\varphi^*}}{C_\Lambda} + 1\right)\eta_{\mu_{k+1}}(\tilde{\varphi}_{\mu_{k+1}}^k) \\
&\leq \left(\frac{C_{\varphi^*}}{C_\Lambda} + 1\right) C_\Lambda \cdot \operatorname{dist}(X^k, \{\varphi_{\mu_{k+1}}^*(T)\}) \\
&= (C_{\varphi^*} + C_\Lambda) \cdot \operatorname{dist}(X^k, \{\varphi_{\mu_{k+1}}^*(T)\}) \\
&= \frac{1}{\gamma} \cdot \operatorname{dist}(X^k, \{\varphi_{\mu_{k+1}}^*(T)\}).
\end{aligned}$$

Here, we used that $\tilde{\varepsilon} < \eta_{\mu_{k+1}}(\tilde{\varphi}_{\mu_{k+1}}^k)$ since the termination criterion is not fulfilled. This proves the inequality in (22) by multiplying both sides by $\gamma > 0$, since the parameter $\mu \in \mathcal{P}$ was arbitrary. Consequently, Algorithm 1 is a weak greedy algorithm.

Regarding the approximation error estimate, we obtain similarly as above that for all $\mu \in \mathcal{P}$ it holds

$$\begin{aligned}
\operatorname{dist}(X^N, \{\varphi_\mu^*(T)\}) &\leq \frac{C_{\varphi^*}}{C_\Lambda}\tilde{\varepsilon} + \eta_{\mu_{N+1}}(\tilde{\varphi}_{\mu_{N+1}}^N) \\
&\leq \left(\frac{C_{\varphi^*}}{C_\Lambda} + 1\right)\tilde{\varepsilon} \\
&= \frac{\tilde{\varepsilon}}{C_\Lambda \gamma} \\
&= \varepsilon,
\end{aligned}$$

where we used that $\eta_{\mu_{N+1}}(\tilde{\varphi}_{\mu_{N+1}}^k) \leq \tilde{\varepsilon}$, *i.e.* that the termination criterion is fulfilled by the last computed final time adjoint. This proves the estimate in (21). □

Based on the last two results, the introduced Algorithm 1 preserves the optimal decay rates, expressed through the Kolmogorov N-widths, of the optimal final time adjoints manifold $\mathcal{M}$. More precisely, the sequences $(d_n(\mathcal{M}))_{n\in\mathbb{N}}$ and $(\sigma_n(\mathcal{M}))_{n\in\mathbb{N}}$, defined by (19) and (20), respectively, exhibit the same asymptotic behaviour, up to a multiplicative constant. As the solution manifold $\mathcal{M}$ is usually beyond our disposal, it is hard to estimate its Kolmogorov N-widths directly. However, these can be related to the Kolmogorov widths of the set of admissible parameters $\mathcal{P}$, which can often be easily determined, even in the case of infinite dimensional parameters (*cf.* [5], Sect. 4.3). More precisely the following result holds [6], under an additional holomorphic assumption.

Theorem 3.8 ([6], Thm. 1.1)**.** *For a pair of complex Banach spaces X and V assume that $u\colon O \to V$ is a holomorphic map from an open set $O \subset X$ into V with uniform bound. If $K \subset O$ is a compact subset of X then for any $\alpha > 1$ and $C_0 > 0$ it holds*

$$d_N(K) \leq C_0 N^{-\alpha} \quad \Longrightarrow \quad d_N(u(K)) \leq C_1 N^{-\beta} \qquad \text{for all } N \in \mathbb{N},$$

for any $\beta < \alpha - 1$ and a constant C_1 depending on C_0, α and the mapping u.

The proof of the last theorem also provides an explicit estimate of the constant C_1 in dependence on C_0, α and the mapping u. Combining this result with Theorems 3.6 and 3.7, one is able to *a priori* estimate the number of snapshots, *i.e.* the dimension of the reduced basis space constructed by the offline part of the greedy algorithm, required to approximate the solution manifold within a given error. However, the price of such an estimate is the holomorphic dependence of the solutions on the parameters, which is unlikely to occur in real-world problems. Moreover, such estimates tend to be conservative and are not often used in practice. For this reason we do not explore Theorem 3.8 and do not require additional smoothness of our parameter mappings, other than the one postulated by Assumption 2.1. We rather run the offline part of the greedy algorithm and observe the dimension of the reduced basis *a posteriori*.

Summarizing, we have seen in this section that Algorithm 1 is a weak greedy algorithm which enables the construction of reduced spaces with theoretically guaranteed convergence properties and error bounds.

3.4. Efficient and reduced online computations

After constructing a reduced basis $\Phi^N \subset X$ (where the dimension $N = |\Phi^N|$ of the reduced space depends on the prescribed error tolerance $\varepsilon > 0$) during the offline phase, the reduced order model can be used during the online phase to replace the costly solution of the original optimal control problem. For a new given parameter $\mu \in \mathcal{P}$, the approximate final time adjoint $\tilde{\varphi}_\mu^N \in X^N = \operatorname{span}\left(\Phi^N\right)$ is computed similarly as described in Section 3.2, see in particular equation (15). For completeness, we state the online procedure also in form of a pseudocode, see Algorithm 3.

Algorithm 3. Online evaluation of the approximate control based on the greedy procedure

Input: Parameter $\mu \in \mathcal{P}$, reduced basis $\Phi^N = \{\varphi_1, \dots, \varphi_N\}$ constructed by Algorithm 1
Output: Approximate final time adjoint $\tilde{\varphi}_\mu^N \in X^N$, approximate optimal control $\tilde{u}_\mu^N \in G$

1: compute $x_i^\mu \leftarrow (I + M\Lambda_\mu^R)\varphi_i$ for $i = 1, \dots, N$
2: assemble operator $\bar{X}_\mu \leftarrow [x_1^\mu \ \cdots \ x_N^\mu] \in \mathcal{L}(\mathbb{R}^N, X)$ for projection onto Y_μ^N
3: compute coefficients $\alpha^\mu = (\alpha_1^\mu, \dots, \alpha_N^\mu)^\top \in \mathbb{R}^N$ as solution of the $N \times N$ linear system (see Eq. (14))

$$\bar{X}_\mu^* \bar{X}_\mu \alpha^\mu = \bar{X}_\mu^* M \left(e^{A_\mu{}^T} x_\mu^0 - x_\mu^T \right) \tag{23}$$

4: compute final time adjoint $\tilde{\varphi}_\mu^N \leftarrow \sum_{i=1}^N \alpha_i^\mu \varphi_i$ (see Eq. (15))
5: solve $-\dot{\tilde{\varphi}}_\mu(t) = A_\mu^* \tilde{\varphi}_\mu(t)$ for $t \in [0, T]$, $\tilde{\varphi}_\mu(T) = \tilde{\varphi}_\mu^N$ backwards in time (see Eq. (5))
6: compute associated control $\tilde{u}_\mu^N(t) \leftarrow -R^{-1} B_\mu^* \tilde{\varphi}_\mu(t)$ (see Eq. (6))
7: **return** $\tilde{\varphi}_\mu^N$, $\tilde{u}_\mu^N$

By means of the error estimator η_μ defined in equation (11), it is possible to obtain an estimation of the error during the online phase as well. By Theorem 3.1, it holds

$$\left\| \varphi_\mu^*(T) - \tilde{\varphi}_\mu^N \right\| \leq \eta_\mu(\tilde{\varphi}_\mu^N).$$

Additionally, if the reduced basis Φ^N is constructed using the greedy algorithm introduced in Section 3.2 for a tolerance $\varepsilon > 0$, we obtain by combining Theorems 3.2 and 3.7 the estimate

$$\eta_\mu(\tilde{\varphi}_\mu^N) \leq C_\Lambda \varepsilon \qquad \text{for all } \mu \in \mathcal{P},$$

and hence it holds

$$\left\| \varphi_\mu^*(T) - \tilde{\varphi}_\mu^N \right\| \leq C_\Lambda \varepsilon \tag{24}$$

for all parameters $\mu \in \mathcal{P}$. We can therefore also bound the error in the online phase given that the reduced space was created with a certain error tolerance.

Since we are typically interested in an approximation of the control u_μ^*, we also give an error bound for the control. To this end, let us define for $t \in [0,T]$ the constant

$$\zeta_{\mu,t} := \left\| R^{-1} B_\mu^* e^{A_\mu^*(T-t)} \right\|_{\mathcal{L}(X,U)} < \infty.$$

Then it holds for all $t \in [0,T]$ that

$$\left\| u_\mu^*(t) - \tilde{u}_\mu^N(t) \right\|_U \leq \zeta_{\mu,t} \left\| \varphi_\mu^*(T) - \tilde{\varphi}_\mu^N \right\|_X .$$

Similar to the bound on the error in the final time adjoint, we can deduce that if the reduced basis Φ^N is constructed with a greedy tolerance of $\varepsilon > 0$, the error in the control can be bounded by

$$\left\| u_\mu^*(t) - \tilde{u}_\mu^N(t) \right\|_U \leq \zeta_{\mu,t} C_\Lambda \varepsilon.$$

The constant $\zeta_{\mu,t}$ depends in particular on the parameter μ and properties of the system matrices, such as stability of the system. Certainly, due to Assumption 2.1 and the compactness of $\mathcal{P}$, there is also a finite uniform upper bound of the constants $\zeta_{\mu,t}$ over the parameter set $\mathcal{P}$.

In our numerical experiments below, we will refer to the reduced order model from Algorithm 3 as G-ROM (greedy reduced order model).

4. Acceleration of the online phase using machine learning

The reduced order model introduced in Section 3 already gives a speedup in solving the optimal control problem compared to solving the exact problem from equation (4), see also the numerical experiments in Section 6. However, the computational costs during the online phase still involve high-dimensional computations. If for instance the state space is finite-dimensional, $X = \mathbb{R}^n$, the costs scale with the dimension n. To be more specific, for a new parameter $\mu \in \mathcal{P}$, one has to compute $x_i^\mu = (I + M\Lambda_\mu^R)\varphi_i$ for $i = 1, \dots, N$ (with N denoting the dimension of the reduced space) by solving the system from equation (10), see also Theorem 2.7. In order to circumvent these steps, we propose to apply machine learning algorithms to directly approximate the map from the parameters to the reduced coefficients. Training data for the machine learning process can be generated using the G-ROM introduced in Section 3. The idea is motivated by a similar approach that has been introduced in the context of parametrized PDEs in [21] and was applied to instationary problems in [46]. In the following section, we first describe the main ideas of our approach in detail. Afterwards, we discuss how the error estimator introduced in Section 3.2 can be applied to also evaluate the error of the machine learning approximation. We finally introduce specific machine learning algorithms used in our numerical experiments below in Section 6.

4.1. Learning the map from parameters to coefficients

During the online phase, each evaluation of the reduced model as shown in Section 3.4 requires the computation of N final time states $x_i^\mu \in X$ in Line 1 of Algorithm 3. Each of these computations amounts in solving a decoupled system of ordinary differential equations, one forward and one backward, as in equations (5) and (7). However, this is only required to setup a (small) linear system of equations to compute the reduced coefficients $\alpha^\mu \in \mathbb{R}^N$ in Line 3 of Algorithm 3. These coefficients determine the projection of the "target" $M(e^{A_\mu T} x_\mu^0 - x_\mu^T)$ to the space of final states, perturbed by the operator $-\left((\Lambda_\mu^R)^{-1} + M\right)$, reachable from the reduced space X^N. Afterwards, in order to compute the approximate control, one only has to solve a backward evolutionary equation (Line 5 in Algorithm 3) and apply the operator $-R^{-1}B_\mu^*$. The main computational effort in Algorithm 3 is spent in Line 1. The idea to accelerate the online computations is to replace steps 1–3 by applying a cheaply to evaluate machine learning surrogate that provides, given a parameter $\mu \in \mathcal{P}$,

an approximation $\hat{\alpha}^\mu \in \mathbb{R}^N$ of the reduced coefficients $\alpha^\mu \in \mathbb{R}^N$ for the parameter μ. Similarly to the remaining steps of Algorithm 3, this results in an approximation $\hat{\varphi}_\mu^N = \sum_{i=1}^N \hat{\alpha}_i^\mu \varphi_i \in X^N$ of the final time adjoint and associated control $\hat{u}_\mu^N \in G$. In other words, we define the mapping $\pi_N \colon \mathcal{P} \to \mathbb{R}^N$ for a parameter $\mu \in \mathcal{P}$ as $\pi_N(\mu) \coloneqq \alpha^\mu$, where $\alpha^\mu \in \mathbb{R}^N$ solves the linear system from equation (23). Furthermore, we construct an approximation $\hat{\pi}_N \colon \mathcal{P} \to \mathbb{R}^N$ of π_N by training a machine learning algorithm and define the machine learning final time adjoint $\hat{\varphi}_\mu^N \approx \tilde{\varphi}_\mu^N$ as

$$\hat{\varphi}_\mu^N \coloneqq \sum_{i=1}^N \big[\hat{\pi}_N(\mu)\big]_i \varphi_i = \sum_{i=1}^N \hat{\alpha}_i^\mu \varphi_i. \tag{25}$$

In our algorithm we first run the greedy algorithm to construct a reduced basis Φ^N. Additionally, the greedy algorithm already returns the pairs $(\mu, \pi_N(\mu)) \in \mathcal{P}_{\text{train}} \times \mathbb{R}^N$ for all $\mu \in \mathcal{P}_{\text{train}}$. Any supervised machine learning algorithm that builds an approximation by using training data, consisting of inputs and corresponding outputs of the function to approximate, can be used in our setting and trained on the data set $D_{\text{train}} = \{(\mu, \pi_N(\mu)) : \mu \in \mathcal{P}_{\text{train}}\}$. In particular, the approach is not restricted to the machine learning methods applied in our numerical experiments. Furthermore, the error estimates presented in the next section are also independent of the exact implementation of the machine learning surrogate.

Remark 4.1 (Smoothness of the parameter to coefficient map)**.** The ability of machine learning algorithms to provide a satisfactory approximation of a function is typically linked to the smoothness of the function. Usually, one can observe that the smoother the function to approximate, the better the machine learning approximation. In our setting, the smoothness of the map π_N depends on the smoothness of the maps $\mu \mapsto A_\mu$, $\mu \mapsto B_\mu$, $\mu \mapsto x_\mu^0$, and $\mu \mapsto x_\mu^T$, *i.e.* all parameter-dependent parts of the optimal control problem. Due to Assumption 2.1, we can also expect a smooth parameter to coefficients map π_N that is amenable to approximation by machine learning surrogates.

For completeness and later reference, we summarize the offline and the online procedure for the machine learning reduced model in the following two pseudocodes:

Algorithm 4. Offline phase of the machine learning greedy procedure.

Input: Greedy tolerance $\varepsilon > 0$
Output: Reduced basis $\Phi^N \subset X$, approximation $\hat{\pi}_N$ of the parameter to coefficients map
1: $\Phi^N, X^N, D_{\text{train}} \leftarrow \textsc{Greedy}(\varepsilon)$ (see Algorithm 1)
2: train a machine learning algorithm using the data D_{train} to obtain a surrogate $\hat{\pi}_N$
3: **return** Φ^N, $\hat{\pi}_N$

Algorithm 5. Online evaluation of the approximate control based on the machine learning greedy procedure.

Input: Parameter $\mu \in \mathcal{P}$, reduced basis Φ^N of size $N = |\Phi^N|$, machine learning approximation $\hat{\pi}_N$ of the parameter to coefficients map
Output: Approximate final time adjoint $\hat{\varphi}_\mu^N \in X^N$, approximate optimal control $\hat{u}_\mu^N \in G$
1: evaluate surrogate $\hat{\pi}_N$ to obtain approximate coefficients $\hat{\alpha}^\mu \leftarrow \hat{\pi}_N(\mu)$
2: compute final time adjoint $\hat{\varphi}_\mu^N \leftarrow \sum_{i=1}^N \hat{\alpha}_i^\mu \varphi_i$ (see Eq. (25))
3: solve $-\dot{\hat{\varphi}}_\mu(t) = A_\mu^* \hat{\varphi}_\mu(t)$ for $t \in [0,T]$, $\hat{\varphi}_\mu(T) = \hat{\varphi}_\mu^N$ backwards in time (see Eq. (5))
4: compute associated control $\hat{u}_\mu^N(t) \leftarrow -R^{-1} B_\mu^* \hat{\varphi}_\mu(t)$ (see Eq. (6))
5: **return** $\hat{\varphi}_\mu^N$, $\hat{u}_\mu^N$

Remark 4.2 (Choice of training parameters for machine learning)**.** It is also possible to add more training samples, *i.e.* pairs of the form $(\mu, \pi_N(\mu))$ for $\mu \in \mathcal{P}$, to the set of training data for the machine learning algorithm. This additional training data can be generated cheaply by running the online algorithm of the G-ROM, see Algorithm 3. In particular, such an enrichment of the training set can be performed adaptively as well, depending on the accuracy of the machine learning prediction. To this end, the *a posteriori* error estimator introduced in Section 3.1 and described in Section 4.2 for the machine learning surrogate might be used to evaluate the machine learning performance, see also Theorem 4.4.

Instead of learning the optimal adjoint datum (or control) directly as functions of parameter (and time), our approach requires learning a mapping between the (typically low-dimensional) spaces $\mathcal{P}$ and $\mathbb{R}^N$ (where usually also N is small). This will allow, as we shall see below, for a significant reduction of the computational costs, while at the same time providing reliable and accurate approximations.

The training data for the machine learning surrogate is generated for free in the offline phase (see Line 15 of Algorithm 1). Solving the exact optimal control problem to generate training data, for instance for learning the controls directly, would be prohibitively expensive both in computational time and in memory. Furthermore, as we will see below, using the greedy algorithm and the reduced basis as an intermediate step results in *a priori* and *a posteriori* error estimates for the machine learning results. These might not be available when directly learning the controls or adjoints.

4.2. *A priori* and *a posteriori* error estimation of machine learning results

We can derive a simple *a priori* bound for the error in the approximation of the optimal final time adjoint that consists of the online greedy error in equation (24) and the machine learning error in approximating the parameter to coefficients mapping:

Lemma 4.3 (Error bound for the machine learning approximation)**.** *Let $\Phi^N = \{\varphi_1, \dots, \varphi_N\} \subset X$ be a reduced basis constructed using the greedy algorithm from Section 3.2 for an error tolerance $\varepsilon > 0$. Further, let $\bar{\Phi}^N \in \mathcal{L}(\mathbb{R}^N, X)$ be the operator given for the i-th unit vector $e_i \in \mathbb{R}^N$ as $\bar{\Phi}^N e_i = \varphi_i$, $i = 1, \dots, N$. Let $\hat{\pi}_N \colon \mathcal{P} \to \mathbb{R}^N$ be an approximation of the exact parameter to coefficients map $\pi_N \colon \mathcal{P} \to \mathbb{R}^N$. Then it holds*

$$\left\|\varphi_\mu^*(T) - \hat{\varphi}_\mu^N\right\| \leq C_\Lambda \varepsilon + \left\|\bar{\Phi}^N\right\|_{\mathcal{L}(\mathbb{R}^N, X)} \left\|\pi_N(\mu) - \hat{\pi}_N(\mu)\right\|$$

for all parameters $\mu \in \mathcal{P}$.

Proof. Using the bound in equation (24) and the definitions of the operator $\bar{\Phi}^N$ as well as the approximate final time adjoints $\tilde{\varphi}_\mu^N$ and $\hat{\varphi}_\mu^N$, it holds for all $\mu \in \mathcal{P}$ that

$$\begin{aligned}
\left\|\varphi_\mu^*(T) - \hat{\varphi}_\mu^N\right\| &\leq \left\|\varphi_\mu^*(T) - \tilde{\varphi}_\mu^N\right\| + \left\|\tilde{\varphi}_\mu^N - \hat{\varphi}_\mu^N\right\| \\
&\leq C_\Lambda \varepsilon + \left\|\bar{\Phi}^N \alpha^\mu - \bar{\Phi}^N \hat{\alpha}^\mu\right\| \\
&= C_\Lambda \varepsilon + \left\|\bar{\Phi}^N \left(\pi_N(\mu) - \hat{\pi}_N(\mu)\right)\right\| \\
&\leq C_\Lambda \varepsilon + \left\|\bar{\Phi}^N\right\|_{\mathcal{L}(\mathbb{R}^N, X)} \left\|\pi_N(\mu) - \hat{\pi}_N(\mu)\right\|.
\end{aligned}$$

□

The first term in the error bound of Theorem 4.3 corresponds to the greedy approximation error in the offline phase and can be adjusted by the choice of the greedy tolerance ε. The second term measures the error of the machine learning in approximating the map π_N. If one uses orthonormalization during the greedy procedure, it even holds $\left\|\bar{\Phi}^N\right\|_{\mathcal{L}(\mathbb{R}^N, X)} = 1$, which further simplifies the error estimate.

The error of the approximate final time adjoint $\hat{\varphi}_\mu^N \in X^N$ can also be estimated in an *a posteriori* manner by means of the error estimator η_μ defined in equation (11). This results, due to Theorem 3.1, in an efficient and reliable error estimator even for the machine learning results, *i.e.* it holds

$$\left\|\varphi_\mu^*(T) - \hat{\varphi}_\mu^N\right\| \leq \eta_\mu(\hat{\varphi}_\mu^N) \leq \left\|I + M\Lambda_\mu^R\right\|_{\mathcal{L}(X,X)} \left\|\varphi_\mu^*(T) - \hat{\varphi}_\mu^N\right\|$$

for all $\mu \in \mathcal{P}$. The error certification and the guarantees obtained by the greedy procedure, see also Theorem 3.7 and equation (24), are the main advantages of the presented approach compared to learning the optimal control directly as a function of the parameter.

Remark 4.4 (Adaptive and certified surrogate model hierarchy from [18])**.** Recently, a model hierarchy consisting of a full-order model (similar to the exact solution of the parametrized optimal control problem in our setting), a reduced order model (similar to the G-ROM created by the greedy algorithm in Sect. 3), and a machine learning surrogate (similar to the one proposed in this section) was introduced in [18] and further tested with deep kernel models as machine learning surrogates in [49]. Due to the error estimation described above for the G-ROM as well as the machine learning surrogate, the adaptive model hierarchy is also applicable in the setting of parametrized optimal control problems as considered in this contribution. The reduced model in [18] is also constructed by an adaptive procedure depending on the error estimate but without selecting the training parameters a priorily. Instead, the adaptive model is queried for different parameters and the reduced model (and hence also the machine learning surrogate) is automatically built and updated if necessary, *i.e.* if the desired error tolerance is not fulfilled. The adaptive model construction from [18] can therefore be seen as a greedy procedure incorporated into the online phase, and thus shows similarities to the greedy construction of the reduced models considered in this paper.

4.3. Machine learning approaches

In this subsection we shortly introduce the machine learning approaches applied in the numerical experiments in Section 6. As already highlighted above, the whole algorithm is not restricted to these specific choices of methods. We start by defining (deep) neural networks in Section 4.3.1 and afterwards describe kernel methods in Section 4.3.2 and Gaussian process regression in Section 4.3.3. More technical details on the implementation and exact choices of certain parts of the machine learning methods can be found in Section 6.1. For practical implementations, we assume that the parameter set $\mathcal{P}$ is finite-dimensional, *i.e.* $\mathcal{P} \subset \mathbb{R}^p$ for some $p \in \mathbb{N}$.

4.3.1. Deep neural networks

Nowadays, deep neural networks are one of the most popular and widespread machine learning algorithms [31]. They have also been applied to model order reduction [18, 21, 46, 49] and optimal control [27]. In this work, we restrict our attention to a standard class of neural networks, so-called *feedforward neural networks*. This class of neural networks applies an alternating sequence of affine transformations and element-wise nonlinear activation functions to the input [38]. The function defined by the neural network can be tailored to the training data by adjusting the entries of the matrices and vectors in the affine transformations, the so-called *weights* and *biases* of the neural network.

To be more precise, we describe the notion of a feedforward neural network following a formal definition from [38], see also Section 4 of [25]: Let $L \in \mathbb{N}$ denote the number of layers of the neural network and $p = N_0, N_1, \ldots, N_{L-1}, N_L = N \in \mathbb{N}$ the numbers of neurons in each layer. Here, we emphasize that the number of neurons in the input layer $N_0 = p$ and the number of neurons in the output layer $N_L = N$ are chosen such that the corresponding neural network defines a mapping $\mathbb{R}^p \to \mathbb{R}^N$ which is required in our application of approximating the parameter to coefficients map. Furthermore, we consider for $i = 1, \ldots, L$ the weight matrices $W_i \in \mathbb{R}^{N_i \times N_{i-1}}$ and bias vectors $b_i \in \mathbb{R}^{N_i}$ which are collected in the tuple $\theta = \big((W_1, b_1), \ldots, (W_L, b_L)\big)$. To introduce nonlinearity, an activation function $\rho\colon \mathbb{R} \to \mathbb{R}$ is applied component-wise between the affine layers. Let $\rho_n\colon \mathbb{R}^n \to \mathbb{R}^n$ denote the component-wise application of ρ to n-dimensional vectors. With these ingredients

at hand, we define the neural network $\Phi_\theta\colon \mathbb{R}^p \to \mathbb{R}^N$ associated to the weights and biases θ for an input $x \in \mathbb{R}^p$ as

$$\Phi_\theta(x) \coloneqq r_L(x),$$

where $r_L\colon \mathbb{R}^p \to \mathbb{R}^N$ is defined recursively by

$$\begin{aligned} r_L(x) &\coloneqq W_L r_{L-1}(x) + b_L, \\ r_i(x) &\coloneqq \rho_{N_i}(W_i r_{i-1}(x) + b_i) \qquad \text{for } i = 1, \dots, L-1, \\ r_0(x) &\coloneqq x. \end{aligned}$$

For a set of training data $(x_i, y_i) \in \mathbb{R}^p \times \mathbb{R}^N$, $i = 1, \dots, n_{\text{train}}$, the weights and biases θ of the neural network are optimized such that the loss function

$$\mathcal{L}_{\text{dnn}}(\theta) \coloneqq \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \|\Phi_\theta(x_i) - y_i\|_2^2$$

is minimized, *i.e.* the mean squared error of the neural network in predicting the outputs y_i given the inputs x_i for $i = 1, \dots, n_{\text{train}}$. Hence, the approximate mapping is chosen as $\hat{\pi}_N \coloneqq \Phi_{\theta^*}$ with $\theta^* \in \operatorname{argmin}_\theta \mathcal{L}_{\text{dnn}}(\theta)$. The loss function $\mathcal{L}_{\text{dnn}}$ is typically optimized using gradient based methods. In our numerical experiments, we apply the quasi-Newton method L-BFGS, see [32]. The gradient of $\mathcal{L}_{\text{dnn}}$ with respect to θ can be efficiently computed *via* backpropagation [42]. To avoid overfitting of the training data, one usually also evaluates the loss function for a validation set, that is chosen distinct from the training set, in each iteration of the optimization algorithm. If the loss on the validation set starts to increase over a couple of consecutive iterations, the optimization is cancelled. This procedure is known as *early stopping*, see [40] for a discussion of different early stopping approaches and [35] for theoretical guarantees in a simplified setting.

The outputs in our application are the coefficients with respect to the reduced basis of final time adjoints. Due to the construction of the reduced basis by the greedy algorithm from Section 3.2, the basis functions are sorted by importance (similar to the sorting of singular vectors in a singular value decomposition). Therefore, coefficients associated to different basis functions vary quite heavily in their magnitude. To ensure a sufficiently accurate approximation of all coefficients, we thus apply a simple scaling to the coefficients before training the neural network. The scaling is an affine transformation taking into account the minimum and maximum of the respective coefficient over the training set such that each coefficient is mapped to the interval $[0, 1]$ (on the training set).

A feedforward neural network with $L \geq 3$ layers is typically called *deep neural network* (DNN). The reduced order model that uses deep neural networks for the coefficient prediction will be called DNN-ROM in the remainder of this paper.

4.3.2. Kernel methods

Approximations using kernel functions have been applied in the context of surrogate modeling quite successfully in the last years, see for instance [18, 44, 49]. Very recently, kernel methods were also used for solving optimal control problems, see [12]. We refer for instance to [47] for a general introduction to kernel methods.

Essentially, these methods build around the notion of (scalar) positive-definite kernels which are mappings $k\colon \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ such that the kernel matrix $[k(x_i, x_j)]_{i,j=1}^n$ is symmetric and positive-definite for all $n \in \mathbb{N}$ and distinct $x_i \in \mathbb{R}^p$, $i = 1, \dots, n$. Associated to every positive-definite kernel k is a reproducing kernel Hilbert space H_k defined as follows, see the Moore–Aronszajn theorem [1]: Let $H_k^0 \coloneqq \operatorname{span}(\{k(\cdot, x) : x \in \mathbb{R}^p\})$, then we can define an inner product $\langle \cdot, \cdot \rangle_{H_k}$ on H_k^0 by

$$\left\langle \sum_{i=1}^n \alpha_i k(\cdot, x_i), \sum_{j=1}^m \beta_j k(\cdot, z_j) \right\rangle_{H_k} \coloneqq \sum_{i=1}^n \sum_{j=1}^m \alpha_i \beta_j k(x_i, z_j)$$

for $n, m \in \mathbb{N}$, $\alpha_i, \beta_j \in \mathbb{R}$ and $x_i, z_j \in \mathbb{R}^p$ for $i = 1, \dots, n$ and $j = 1, \dots, m$. The space H_k is now given as the completion of H_k^0 with respect to the inner product $\langle \cdot, \cdot \rangle_{H_k}$, *i.e.* it holds $H_k = \overline{H_k^0}^{\langle \cdot, \cdot \rangle_{H_k}}$. Furthermore, the space H_k has the function k as reproducing kernel, *i.e.* it holds $\Phi(x) = \langle \Phi, k(\cdot, x) \rangle_{H_k}$ for all $\Phi \in H_k$ and $x \in \mathbb{R}^p$. Given a set of data points $(x_i, y_i) \in \mathbb{R}^p \times \mathbb{R}$, $i = 1, \dots, n_{\text{train}}$, kernel methods typically try to minimize a loss function $\mathcal{L}_{\text{kernel}} \colon H_k \to \mathbb{R}$ (similar to the loss function $\mathcal{L}_{\text{dnn}}$ introduced above for the training of neural networks) of the form

$$\mathcal{L}_{\text{kernel}}(\Phi) \coloneqq \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} |\Phi(x_i) - y_i|^2 + \lambda \|\Phi\|_{H_k}^2$$

for $\Phi \in H_k$, where $\lambda \geq 0$ denotes a regularization parameter and $\|\cdot\|_{H_k} \coloneqq \sqrt{\langle \cdot, \cdot \rangle_{H_k}}$ is the norm induced by the inner product $\langle \cdot, \cdot \rangle_{H_k}$ on the reproducing kernel Hilbert space. A well-known representer theorem, see for instance [4], reveals that there are coefficients $\alpha_i \in \mathbb{R}$, $i = 1, \dots, n_{\text{train}}$, such that it holds

$$\Phi^* = \sum_{i=1}^{n_{\text{train}}} \alpha_i k(\cdot, x_i),$$

where $\Phi^* = \operatorname{argmin}_{\Phi \in H_k} \mathcal{L}_{\text{kernel}}(\Phi)$ minimizes the loss function. In particular, the minimizer of the loss function can be represented as a linear combination of the kernel k evaluated only at the data points x_i, $i = 1, \dots, n_{\text{train}}$.

The setting introduced above can be extended to vector-valued outputs $y_i \in \mathbb{R}^N$ as required in our use case of approximating the coefficients of the reduced representation of the optimal final time adjoint. To this end, one considers matrix-valued kernels $k_N \colon \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}^{N \times N}$ defined as $k_N = k \cdot I_N$ and vector-valued coefficients $\alpha_i \in \mathbb{R}^N$, where $I_N \in \mathbb{R}^{N \times N}$ denotes the $N \times N$ identity matrix. Furthermore, to obtain a surrogate that can be evaluated fast, it is beneficial to obtain a sparse approximation of Φ^*. To be more precise, one aims to select an appropriately chosen subset $\Xi \subset \{1, \dots, n_{\text{train}}\}$ of the training set such that $|\Xi| \ll n_{\text{train}}$ and approximate Φ^* as

$$\Phi^* \approx \hat{\Phi} \coloneqq \sum_{i \in \Xi} \alpha_i k_N(\cdot, x_i) \tag{26}$$

with certain coefficients $\alpha_i \in \mathbb{R}^N$ for $i \in \Xi$. The selection of the training inputs used in the approximation, *i.e.* of the set Ξ, can for instance be done using a greedy algorithm similar to those described in Section 3. One very popular example of such an algorithm is the *vectorial kernel orthogonal greedy algorithm* (VKOGA), see [44, 48], that is used for our numerical experiments below.

We will refer to the surrogate model using $\hat{\pi}_N \coloneqq \hat{\Phi}$ built by the VKOGA algorithm as VKOGA-ROM.

4.3.3. Gaussian process regression

Applied to regression and probabilistic classification tasks, Gaussian processes are an indispensable tool in the machine learning landscape. In particular in the context of regression problems, thus known as *Gaussian process regression* (GPR), these methods have proven to provide satisfactory results in many applications. Gaussian process regression has recently been applied to reduced order modeling in [16], and also to (stochastic) optimal control in [33]. For a general introduction to Gaussian processes in the context of machine learning and in particular to Gaussian process regression, see [41]. There are strong connections between kernel methods and Gaussian process regression, see for instance [24] for a review concerning this matter. The same topic is also discussed in Chapter 2.2 of [41]. Our presentation in this section follows [41].

Gaussian process regression starts from a parametrized model function whose parameters are determined such that given training data are likely to occur as results of the model and that the chosen parameters also have a high probability. To make this precise, let us denote by θ the parameters, also known as *weights*, of our model function. A probability distribution is assigned to these parameters that determines how likely certain values of the parameters are to occur in the model. This distribution is called *prior* and denoted as $P(\theta)$. Furthermore, we collect the inputs of our training set in a matrix $X \in \mathbb{R}^{n_{\text{train}} \times p}$ and the corresponding outputs

in a matrix $Y \in \mathbb{R}^{n_{\text{train}} \times N}$. Furthermore, we denote by $P(Y|X,\theta)$ the so-called *likelihood* which corresponds to the probability of the outputs Y given the weights θ. We should emphasize at this point that the output Y is assumed to be affected by random noise and hence it is given as a random disturbance of the true model with unknown weights θ. The *posterior* distribution of the weights $P(\theta|Y,X)$ can now be computed using Bayes' rule as

$$P(\theta|Y,X) = \frac{P(Y|X,\theta)P(\theta)}{P(Y|X)}.$$

Given a new input $x \in \mathbb{R}^p$, the prediction of the model is a probability distribution $P(y|x,X,Y)$ which for a possible output $y \in \mathbb{R}^N$ is calculated as

$$P(y|x,X,Y) = \int P(y|x,\theta)P(\theta|X,Y)\,\mathrm{d}\theta.$$

It provides the average of the likelihood for the individual sample $(x,y) \in \mathbb{R}^p \times \mathbb{R}^N$ over the possible parameters weighted by the posterior distribution given the weights θ. As the final prediction of the model, the mean $\mathbb{E}_y[P(y|x,X,Y)]$ of the distribution $P(y|x,X,Y)$ is taken. Additional properties of the distribution $P(y|x,X,Y)$, such as for instance its variance or higher order moments, can be used to further characterize the result and obtain confidence intervals for the prediction. Typically, the prior is defined by a covariance function in form of a kernel similar to the ones introduced in the previous section on kernel methods. The hyper-parameters of the kernel are fitted during the construction of the surrogate, see [41] for more details.

The reduced model based on Gaussian process regression, which considers the approximate mapping from parameter to coefficients defined as $\hat{\pi}_N(x) \coloneqq \mathbb{E}_y[P(y|x,X,Y)]$, will be called GPR-ROM in the following.

5. Comparison of computational costs

In this section we compare the overall computational costs during the offline and the online phase of the three methods considered in this contribution: Solving the linear system from (10) for the optimal final time adjoint $\varphi_\mu^*(T)$, computing the approximation $\tilde{\varphi}_\mu^N$ using Algorithm 3, and computing the machine learning surrogate solution $\hat{\varphi}_\mu^N$ as described in Section 4. Additionally, we also consider the computational costs for evaluating the error estimator $\eta_\mu(p)$ for a given approximate final time adjoint $p \in X$. The error sestimator can be applied to both reduced models provided by Algorithms 3 and 5 and might be helpful to certify the results *a posteriori* (see also Thm. 4.4). It should as well be stressed at the beginning of the section that all costs are stated for the finite-dimensional setting, *i.e.* we assume here $n \coloneqq \dim(X) < \infty$.

The overall assumption is that the reduced basis constructed by the greedy algorithm is small compared to the dimension of the state space, *i.e.* that it holds $N \ll n$. In this case, we will see below that the reduced models considered in this contribution can greatly reduce the computational effort compared to the exact solution of the optimal control problem. Due to the error estimates presented in Sections 3.3, 3.4 and 4.2, the reduced models can achieve accurate approximations at the same time as well.

Below we first consider general assumptions on the discretization of the optimal control system and the costs for computing certain matrix decompositions that help reducing the overall computational effort. Afterwards, we continue by discussing the online costs and the costs for the *a posteriori* error estimator, because some of these algorithms appear as steps in the offline procedures as well. Finally, we state the costs for the offline computation of the considered reduced models.

5.1. Preprocessing costs

In the following, we assume that a temporal discretization consisting of $n_t \in \mathbb{N}$ time steps is used for each ordinary differential equation that arises in the algorithms. This means that we use a time step size of $\Delta t = T/n_t$. Furthermore, we use implicit time stepping methods, such as the well-known Crank–Nicolson scheme [7], which require the solution of linear systems of equations in each time step. Since the system matrices

arising in our setting are assumed to be independent of time, all linear systems of equations involve the same fixed matrices in each time step (for a fixed parameter $\mu \in \mathcal{P}$). We can thus first compute the lower–upper (LU) decomposition Chapter 3.2 of [13] of A_μ and afterwards solve linear systems involving A_μ by forward and backward elimination. Similarly, since the matrix R is symmetric and positive-definite, it possesses a Cholesky decomposition Chapter 4.2.3 of [13], *i.e.* we can write $R = \tilde{R}\tilde{R}^\top$ where $\tilde{R} \in \mathbb{R}^{m\times m}$ is a lower triangular matrix. Using the Cholesky decomposition, one can replace the costly solution of a dense linear system by forward and backward elimination, using that the Cholesky factor $\tilde{R}$ is a lower triangular matrix. Hence, the Cholesky decomposition of R can be used to speedup the computation of u_μ, which involves solving multiple linear systems of equations with R as system matrix (see Eq. (4a)). Certainly, in the case of a parameter-dependent weighting matrix R_μ, it is generally impossible to precompute the Cholesky factor of the weighting matrix since it changes with the parameter.

Computing the LU decomposition of $A_\mu \in \mathbb{R}^{n\times n}$ requires $\mathcal{O}(n^3)$ operations in general. For the Cholesky decomposition of $R \in \mathbb{R}^{m\times m}$, in total $\mathcal{O}(m^3)$ operations are needed. The preprocessing costs therefore amount to

$$\mathcal{O}\big(n^3 + m^3\big) \tag{27}$$

operations. To solve an ordinary differential equation of the form as in equation (1) or in equation (4a), we need $\mathcal{O}\big(n_t n(n+m)\big)$ operations under the assumption that the respective LU decompositions and Cholesky decompositions have already been computed before. For systems with sparse system matrices, for instance those stemming from finite element discretizations of PDEs, it might nevertheless be beneficial to use iterative solvers instead of LU decompositions.

5.2. Costs for calculating the (approximate) optimal control

In the next three subsections we discuss the costs for computing the exact optimal control, its approximation using Algorithm 3 and the machine learning based approximation from Algorithm 5. We already emphasize at this point that the costs stated below will contain the costs for computing the (approximate) optimal adjoint datum as well as the costs for deriving the (approximate) optimal control from the adjoint datum. Furthermore, we note that in typical applications it holds $n_t \geq n$ and $m \ll n$.

5.2.1. Exact optimal control

In order to compute the exact optimal final time adjoint $\varphi_\mu^*(T)$ (up to a prescribed numerical tolerance), one has to solve the linear system stated in Theorem 2.5. Recall that M and Λ_μ^R are symmetric and positive semi-definite matrices. If they commute, also $I + M\Lambda_\mu^R$ is symmetric and positive semi-definite. Consequently, the linear system of equations in (10) can be solved efficiently for $\varphi_\mu^*(T)$ using the conjugate gradient (CG) method ([13], Chap. 10.2). The CG method only requires applications of the (symmetric and positive-definite) system matrix to certain vectors. As described in Theorem 2.7, this can be done in our case without explicitly constructing the Gramian matrix Λ_μ^R. However, it is still required to solve two initial value problems (one for the adjoint state and one for the primal state) to compute a single product $(I + M\Lambda_\mu^R)p$ for a vector $p \in X$. This step can therefore become costly for large systems and needs to be performed multiple times in each iteration of the CG algorithm. We also emphasize that the system matrix $I + M\Lambda_\mu^R$ depends on the parameter $\mu \in \mathcal{P}$ and can thus not be efficiently precomputed (even under assumptions such as affine parameter-dependence of A_μ and B_μ, due to the matrix exponential in Eq. (8)). The CG algorithm requires to solve the optimality system in equation (4) multiple times for different terminal conditions for the adjoint state. The number of iterations of the CG algorithm depends on the condition number of the system matrix at hand. To enable a comparison with the reduced models below, let us denote by $n_{\mathrm{CG}} \in \mathbb{N}$, $n_{\mathrm{CG}} \leq n$, the number of iterations needed in the CG algorithm until the norm of the residual drops below a prescribed tolerance (we omit at this point an explicit dependence of the number of iterations on the parameter μ, but instead assume that the number of

iterations is similar for all parameters $\mu \in \mathcal{P}$). Computation of the optimal control u_μ^* for a parameter $\mu \in \mathcal{P}$ requires

$$\mathcal{O}\big(n^3 + m^3 + n_{\mathrm{CG}} n_t n(n+m)\big)$$

operations, where also the preprocessing costs stated in equation (27) are considered.

Remark 5.1. If $M\Lambda_\mu^R$ is not symmetric and positive semi-definite (or if no guarantees on it are given *a priori*), the linear system in Theorem 2.5 can be solved by other iterative methods. For instance, gradient descent can be used on the least-squares problem. Again, the computation of the full Gramian is not required, but only its application on vectors. The cost per each iteration is then the cost of the resolution of a forward and a backward dynamical system in dimension n with n_t time steps. Moreover, for a given tolerance, the number of iterations needed to achieve that precision is given by the convergence rates of the method. In what follows, for simplicity and due to the clearness of the complexity of the CG method, we stick to the case in which the CG method can be applied.

5.2.2. Reduced optimal control based on the greedy procedure

In order to compute the reduced optimal adjoint $\tilde{\varphi}_\mu^N$ for a given parameter $\mu \in \mathcal{P}$ according to Algorithm 3, we first compute the decompositions of R and A_μ as already described above. Furthermore, Line 1 in Algorithm 3 requires the solution of $2N$ evolution systems (the primal and the adjoint for each vector of the reduced basis), each of the cost $\mathcal{O}(n_t n(n+m))$. Assembling the matrix $\bar{X}_\mu^\top \bar{X}_\mu \in \mathbb{R}^{N\times N}$ requires $\mathcal{O}(N^2 n)$ operations, while solving the corresponding linear system of equations for the reduced coefficients in equation (23) is of complexity $\mathcal{O}(N^3)$. Computing the linear combination of the basis functions according to equation (15) requires $\mathcal{O}(Nn)$ operations and is hence negligible compared to the complexity for setting up the linear system. Altogether, taking into account the preprocessing costs equation (27), the costs for computing the reduced optimal control using Algorithm 3 is of complexity

$$\mathcal{O}\big(n^3 + m^3 + N n_t n(n+m) + N^2 n + N^3\big).$$

Consequently, the online phase of the greedy reduced order model provides a speedup compared to computing the exact optimal control as long as it holds $N < n_{\mathrm{CG}}$.

5.2.3. Reduced optimal control based on the machine learning greedy procedure

Typically, the evaluation of a machine learning surrogate $\hat{\pi}_N \colon \mathcal{P} \subset \mathbb{R}^p \to \mathbb{R}^N$ requires $\mathcal{O}\big(\chi(p+N)\big)$ operations, where $\chi \colon \mathbb{N} \to \mathbb{N}$ is a polynomial of moderate degree. For instance in the case of deep neural networks, see Section 4.3.1, the numbers of neurons in each layer of the network usually scale with $p+N$ and therefore the computational effort for a forward pass through the neural network is of complexity $\mathcal{O}\big((p+N)^2\big)$ due to the required matrix-vector multiplications. The constants in the bound depend on the number of layers in the network. Together with the reconstruction step of the approximate final time adjoint from the coefficients, which requires $\mathcal{O}(Nn)$ operations, see equation (25), the total costs for the online phase in the case of the machine learning reduced model amount to

$$\mathcal{O}\big(n^3 + m^3 + n_t n(n+m) + \chi(p+N) + Nn\big).$$

We emphasize at this point that the costs for computing the approximate control using the machine learning surrogate is dominated by the computation of the control for the given approximate final time adjoint, which requires $\mathcal{O}\big(n^3 + m^3 + n_t n(n+m)\big)$ operations. The computation of the approximate final time adjoint itself is typically much faster and of complexity $\mathcal{O}\big(\chi(p+N) + Nn\big)$ when applying the machine learning surrogate.

In comparison to the computational costs for the reduced model based on the greedy procedure, see Section 5.2.2, the complexity of evaluating the machine learning based reduced model (under the assumptions that $p \ll n$, $N \ll n$, $m \ll n$, and $n_t \geq n$) is given by $\mathcal{O}\big(n_t n(n+m)\big)$ while the reduced model from Section 3.4 requires $O\big(N n_t n(n+m)\big)$ operations. As we will see in the numerical experiments in Section 6, this reduction in the computational complexity will result in a serious speedup when using the machine learning procedure.

5.3. Evaluation of the *a posteriori* error estimator

Given a parameter $\mu \in \mathcal{P}$ and an approximate final time adjoint $p \in X$, evaluating the error estimate $\eta_\mu(p)$ defined in equation (11) mainly requires the same steps as evaluating the reduced model in Algorithm 3, except that the system in equation (4) has to be solved only once and not N times. Therefore, the dominating costs for the error estimator are of the order

$$\mathcal{O}\big(n^3 + m^3 + n_t n(n+m)\big).$$

We observe in particular that the evaluation of the error estimator is of the same complexity as computing the reduced optimal control using the machine learning model, see Section 5.2.3.

If the approximate final time adjoint $p \in X$ has been computed using the online procedure of the G-ROM from Algorithm 3, *i.e.* if it holds $p = \tilde{\varphi}_\mu^N \in X^N$, we can use that

$$(I + M\Lambda_\mu^R)\tilde{\varphi}_\mu^N = \sum_{i=1}^{N} \alpha_i^\mu (I + M\Lambda_\mu^R)\varphi_i = \sum_{i=1}^{N} \alpha_i^\mu x_i^\mu. \tag{28}$$

Therefore, we can reuse the states x_i^μ that have already been computed in Line 1 and the right hand side of the system given by $M\left(e^{A_\mu T}x_\mu^0 - x_\mu^T\right)$ computed in Line 3 of Algorithm 3 to reduce the computational costs of evaluating the error estimator $\eta_\mu(\tilde{\varphi}_\mu^N)$ to $\mathcal{O}\big(Nn\big)$, which is required for computing $(I + M\Lambda_\mu^R)\tilde{\varphi}_\mu^N$ according to (28) and the norm of the residual. All other components of the residual have already been computed in Algorithm 3. This simplification can also be used in the implementation of the offline greedy procedure in Algorithm 1.

5.4. Costs for the offline computations

Subsequently we state the costs for performing the greedy procedure in Algorithm 1 and the machine learning greedy procedure from Algorithm 4. In the discussion we make use of the results on the costs for the online computations obtained above.

5.4.1. Greedy procedure

If the storage capabilities allow, we can precompute the LU decomposition of A_μ for all parameters $\mu \in \mathcal{P}_{\text{train}}$, which requires $\mathcal{O}(n_{\text{train}}n^3)$ operations. Afterwards, the `while`-loop in Algorithm 1 runs for N iterations (where N is certainly unknown before executing the algorithm). In each iteration, the exact optimal adjoint datum for one training parameter has to be computed (which needs $\mathcal{O}(n_{\text{CG}}n_t n(n+m))$ operations, see Sect. 5.2.1). Furthermore, for each of the n_{train} training parameters the reduced optimal adjoint has to be computed, which is of complexity $\mathcal{O}(Nn_t n(n+m)+N^2n+N^3)$ (since the iteration counter k is increased up to a value of N), and the *a posteriori* error estimator is evaluated (which costs only $\mathcal{O}(n)$ operations, see the discussion in Sect. 5.3). Altogether, the complexity of running Algorithm 1 can be estimated as

$$\mathcal{O}\left(n_{\text{train}}n^3 + m^3 + N\left(n_{\text{CG}}n_t n(n+m) + n_{\text{train}}\big(Nn_t n(n+m) + N^2n + N^3\big)\right)\right).$$

5.4.2. Machine learning greedy procedure

To build a machine learning surrogate using the machine learning greedy procedure from Algorithm 4, the greedy procedure in Algorithm 1 is performed as the first step. Afterwards, the machine learning algorithm is trained using the collected training data D_{train}. If we denote by C_{train} the costs of training the machine learning surrogate (which typically depends on the size n_{train} of the training set, the dimension p of the parameter space, the reduced dimension N, and the polynomial χ introduced in Sect. 5.2.3), the overall costs for Algorithm 4 are of the complexity

$$\mathcal{O}\left(n_{\text{train}}n^3 + m^3 + N\left(n_{\text{CG}}n_t n(n+m) + n_{\text{train}}\big(Nn_t n(n+m) + N^2n + N^3\big)\right) + C_{\text{train}}\right).$$

6. Numerical experiments

In the following section we investigate the proposed algorithms in two numerical examples. We first consider a parametrized version of the heat equation where the parameter influences the conductivity as well as the target state. The control acts on the two boundaries of the one-dimensional computational domain. As a second example, we examine a damped wave equation with boundary control where the parameter changes the wave propagation speed.

At this point, we want to stress that our theory is developed for bounded operators only and as such excludes direct application to PDEs. In the numerical examples below, we instead consider finite-dimensional approximations of these PDEs and their solutions.

Before describing the numerical experiments in detail, we give an overview of some implementational aspects and discuss the software used to perform the test cases.

6.1. Implementational details

The temporal and spatial discretizations of the considered equations are specified below individually for the respective examples. In this section we only state general details that are the same for both experiments.

As discussed in Section 2.2, we apply the CG algorithm to solve the optimality system for the exact optimal final time adjoint. We choose a tolerance of $\varepsilon_{\mathrm{CG}} = 10^{-12}$ for the CG method in all experiments, *i.e.* the algorithm is stopped once the Euclidean norm of the residual drops below $\varepsilon_{\mathrm{CG}}$.

All numerical test cases are implemented in the `Python` programming language. Computations dealing with vectors and matrices, in particular computations of matrix decompositions, use the `numpy` [20] and `scipy` [45] packages. The neural networks are implemented using `PyTorch` [36] and are adapted from the code used in the model order reduction software `pyMOR` [34]. Furthermore, we apply the `VKOGA` library[1] [44] for implementing the kernel methods, and use the implementation of Gaussian process regression available in `scikit-learn` [37].

We use a fixed neural network architecture throughout the experiments that consists of three hidden layers with 50 neurons in each layer, *i.e.* $L = 4$, $N_0 = p$, $N_1 = N_2 = N_3 = 50$, and $N_4 = N$. This architecture is chosen sufficiently large for our application and no special adaptation was necessary in our experiments. As activation function we apply the well-known hyperbolic tangent $\rho = \tanh$. As described in Section 4.3.1 we use the L-BFGS algorithm implemented in `PyTorch` for optimization and perform 10 restarts of the training using random initial weights and biases. The neural network obtaining the smallest loss is finally returned by the training procedure and used for the prediction. For validation of the results during the optimization, 10% of the training parameters are randomly selected as a validation set. The validation data is used to evaluate the early stopping criterion which checks whether the loss decreased within the last 10 optimization steps. If that is not the case, the training is stopped.

For the VKOGA-ROM we apply the Gaussian radial basis function kernel $k\colon \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ defined for $x, y \in \mathbb{R}^p$ as

$$k(x,y) \coloneqq \exp\left(-\left(\beta \left\|x-y\right\|_2\right)^2\right),$$

where the constant $\beta > 0$ is defined for the heat equation experiment in Section 6.2 as $\beta = 0.1$ and for the wave equation example in Section 6.3 as $\beta = 1$. Further, in order to select the subset Ξ of training parameters used for the sparse representation of the kernel interpolant in equation (26), we use the P-greedy algorithm (see [44] for more details) with a tolerance of $\varepsilon_P = 10^{-10}$. The regularization parameter λ is set to $\lambda = 0$, *i.e.* no additional regularization is incorporated in the loss function.

For the Gaussian process regression surrogate, we apply the prior defined by the covariance function given through the kernel k defined as

$$k(x,y) \coloneqq c \cdot \exp\left(-\frac{\|x-y\|_2^2}{2l^2}\right)$$

[1]The `VKOGA` library is available at https://github.com/GabrieleSantin/VKOGA

for inputs x, y with the hyper-parameters $c \in [0.1, 1000]$ and $l \in [0.001, 1000]$. The optimization process for fitting the hyper-parameters is restarted 10 times with different initial guesses to maximize the log-marginal likelihood. To ensure that the matrix appearing during the fitting process is positive-definite, we add a constant of $\alpha = 0.001$ to the diagonal of the kernel matrix. Furthermore, the outputs are normalized to zero mean and unit variance which fits to the chosen prior with the same properties.

Regarding the constants C_Λ, C_{φ^*}, and γ introduced in Section 3.2, we remark that there typically do not exist sharp and easy to compute estimates for these constants. A naive estimate using submultiplicativity of the matrix norm and the triangle inequality results in pessimistic estimates that are impractical in numerical experiments. We therefore directly prescribe the tolerance $\tilde{\varepsilon}$ and select an appropriate training set $\mathcal{P}_{\text{train}} \subset \mathcal{P}$ (instead of specifying the error tolerance ε), which is sufficient to run the greedy procedure from Algorithm 1 starting in Line 4.

The numerical experiments have been carried out on a dual socket compute server with two Intel(R) Xeon(R) Gold 6254 CPUs running at 3.10 GHz and 36 cores in each CPU.

We also provide the source code for our numerical experiments [26], which can be used to reproduce the numerical results stated below[2].

6.2. Test case 1: Heat equation

As a first example we consider the heat equation in one spatial dimension with a two-dimensional parameter $\mu \in \mathcal{P} := [1, 2] \times [0.5, 1.5] \subset \mathbb{R}^2$. The first component μ_1 of the parameter $\mu = [\mu_1, \mu_2] \in \mathcal{P}$ determines the conductivity in the equation whereas the second component μ_2 determines the target state v_μ^T. The problem of interest is given as

$$\begin{aligned} \partial_t v_\mu(t,y) - \mu_1 \Delta v_\mu(t,y) &= 0 && \text{for } t \in [0,T], y \in \Omega, \\ v_\mu(t,0) &= u_{\mu,1}(t) && \text{for } t \in [0,T], \\ v_\mu(t,1) &= u_{\mu,2}(t) && \text{for } t \in [0,T], \\ v_\mu(0,y) &= v_\mu^0(y) = \sin(\pi y) && \text{for } y \in \Omega. \end{aligned}$$

Here, we denote by $u_\mu(t) = \left[u_{\mu,1}(t), u_{\mu,2}(t)\right]^\top \in \mathbb{R}^2$ for $t \in [0,T]$ the (two-dimensional) control that influences the boundary conditions on both ends of the spatial domain $\Omega = [0,1]$. The target state is given as $v_\mu^T(y) = \mu_2 y$ for $y \in \Omega$. Furthermore, we consider the dynamics until the final time $T = 0.1$ is reached. The system above is discretized in space using a second-order central finite difference scheme on a uniform spatial grid consisting of $n_y = 100$ inner points with a spatial grid size of $h = 1/(n_y + 1)$. In such a way the above system can be written in the form of (1) with the system matrices given as

$$A_\mu = \frac{\mu_1}{h^2}\bar{A} \qquad \text{with} \qquad \bar{A} = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix} \in \mathbb{R}^{n \times n} \quad \text{and} \quad B_\mu = \frac{\mu_1}{h^2} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{n \times 2}, \tag{29}$$

where the system dimension is $n = n_y$. This system fulfills the well-known *Kalman rank condition* for all parameters $\mu \in \mathcal{P}$ and is therefore controllable. Furthermore, the continuity requirements stated in Assumption 2.1 are also obviously fulfilled due to the choice of the system matrices A_μ and B_μ and the target state v_μ^T.

For discretization of the time derivative we apply the Crank-Nicolson method, see [7], for $n_t = 30 \cdot n_y$ uniform time steps with a time step size of $\Delta t = 1/n_t$.

In the context of discretized PDEs, the standard Euclidean inner product might not be well-suited. We therefore choose a weighted version of the Euclidean inner product for the state space that takes into account

[2]The corresponding `GitHub`-repository containing the source code is available at https://github.com/HenKlei/ML-OPT-CONTROL

the spatial discretization size h, see [17]. To be more precise, we equip the state space $X = \mathbb{R}^n$ with the inner product $\langle \cdot, \cdot \rangle_h$ defined for $x, y \in X$ as

$$\langle x, y \rangle_h = h \langle x, y \rangle_2, \tag{30}$$

where $\langle \cdot, \cdot \rangle_2$ denotes the standard Euclidean inner product on $\mathbb{R}^n$. For the space of controls $U = \mathbb{R}^2$ we use the standard Euclidean inner product. However, on the time-discretized control space, *i.e.* the discretized version of the space G, we consider the norm $\|\cdot\|_{\Delta t}$ defined for $u = [u_{i,j}]_{i=1,\dots,n_t; j=1,2} \in \mathbb{R}^{n_t \times 2}$ as

$$\|u\|_{\Delta t} \coloneqq \left(\Delta t \sum_{i=1}^{n_t} \|u_i\|_2^2 \right)^{\frac{1}{2}}, \tag{31}$$

which is only used to compare optimal and approximate controls. Due to the definition of the cost functional in equation (2), the inner product $\langle \cdot, \cdot \rangle_h$ also enters the optimal control problem.

The weighting matrices $M \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{2 \times 2}$ are chosen as

$$M = I \qquad \text{and} \qquad R = \begin{bmatrix} 0.125 & 0 \\ 0 & 0.25 \end{bmatrix},$$

which in particular fulfill Assumption 2.6. The different weights for the different components of the control have the effect that the size of the second component $u_{\mu,2}$ is penalized stronger than the size of the first one $u_{\mu,1}$, *i.e.* controlling the right boundary value of the state is considered to be more costly than controlling the left boundary value. An example of the optimal final time adjoint as well as the associated controls and states for the parametrized heat equation is shown in Figure 3 for the parameter $\mu = (1.5, 0.75) \in \mathcal{P}$. We denote the discretized states by x_μ to use the same notation as in the previous sections. In the top left plot of the figure we notice that the initial state is driven close to the target state over time with the control acting on the boundary points. This can also be seen in the bottom left plot of the figure where the initial state, target state and optimal state at final time are compared. Since we are solving an optimal control problem where a deviation from the target state is penalized but the system is not forced to hit the target state exactly, we see a slight deviation from the target state in the optimal final time state. Furthermore, we observe in the top right plot of Figure 3 that the optimal final time adjoint is very smooth, which facilitates the approximation of optimal final time adjoints by lower-dimensional subspaces. The bottom right plot of Figure 3 shows how the two components of the control change over time. In particular at the final time $T = 0.1$, the first component $u^*_{\mu,1}$ is close to 0 while the second component $u^*_{\mu,2}$ is close to the value 0.75, as expected due to the choice of the target state $v^T_\mu(y) = \mu_2 y$, the parameter value $\mu_2 = 0.75$, and because the controls act on the boundary values of the solution.

We apply the greedy procedure from Section 3.2 to construct a reduced basis using $n_{\text{train}} = 64$ samples on a uniform 8×8 grid in the parameter space $\mathcal{P}$. At this point, we emphasize once again that the full optimal control problem is not solved for all parameters in the training set $\mathcal{P}_{\text{train}}$ during Algorithm 1. Instead, the exact solution is only computed for those parameters that are selected by the greedy procedure. It is therefore possible to consider a relatively large training set for the greedy algorithm since the error estimator is rather cheap to evaluate.

The results of the greedy algorithm are depicted in Figure 4. The greedy tolerance was set to $\tilde{\varepsilon} = 10^{-6}$ which is reached for a reduced basis size of $N = 8$, see Figure 4. Since no prior knowledge of the constants C_Λ and γ is available for this example, we directly set the tolerance $\tilde{\varepsilon}$ appearing in the termination criterion in Line 7 of Algorithm 1 instead of the greedy tolerance ε. We also observe that the overestimation of the true error by the error estimator becomes smaller with a larger reduced basis size and that the estimated maximum error is close to the true maximum error. In Figure 5, the singular values of the optimal final time adjoints associated to the $n_{\text{train}} = 64$ training parameters are shown in decreasing order. To be more precise, we solved for the optimal final time adjoint for all training parameters, collected these vectors as columns in a matrix, and computed the singular value decomposition of that matrix. Due to equivalences of norms, the respective singular values show

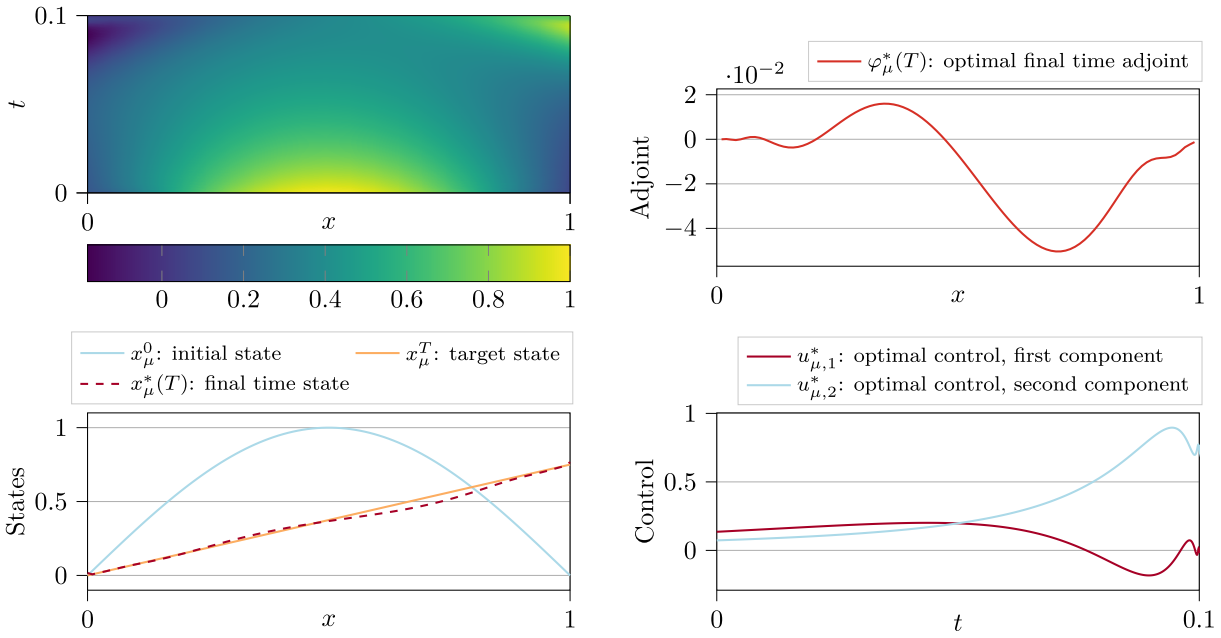

FIGURE 3. Optimal state x^*_μ in a space-time plot (*top left*), optimal final time adjoint $\varphi^*_\mu(T)$ (*top right*), optimal control u^*_μ (*bottom right*) and initial x^0_μ, final $x^*_\mu(T)$ and target x^T_μ states (*bottom left*) for the parameter $\mu = (1.5, 0.75)$ in the heat equation example.

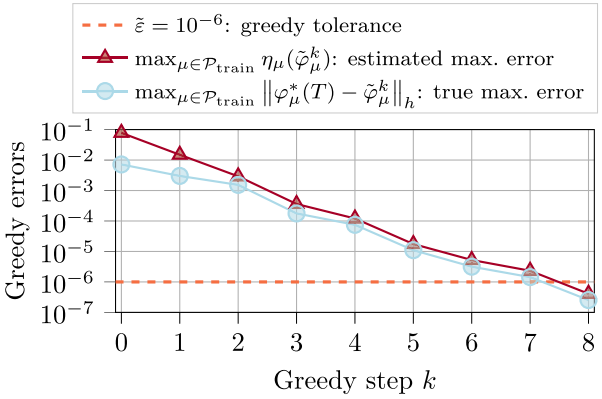

FIGURE 4. Results of the greedy algorithm applied to the heat equation example.

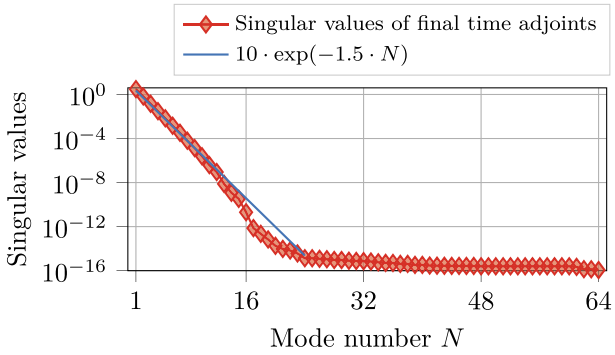

FIGURE 5. Singular value decay of optimal final time adjoints for the heat equation.

(approximately) the order of the decay of the Kolmogorov N-width of the solution manifold $\mathcal{M}$ of optimal final time adjoints (at least on the training set $\mathcal{P}_{\text{train}}$). One can observe an exponential decay in the singular values until machine precision is reached, see also the blue line marked by —— in the figure. This suggests that the manifold $\mathcal{M}$ is amenable for approximation by low-dimensional linear subspaces. Hence, the greedy algorithm can be applied in this setting and finds a suitable reduced space according to Theorems 3.6 and 3.7.

Afterwards, the same set of training parameters as for the greedy algorithm has been used for training the different machine learning surrogates. Once again, we can use the relatively large training set consisting of $n_{\text{train}} = 64$ parameters, because the computation of the training snapshots is already performed during

TABLE 1. Results of the numerical experiments for the heat equation using the G-ROM, DNN-ROM, VKOGA-ROM and GPR-ROM.

Method	Max. error adjoint	Avg. error adjoint	Max. error control	Avg. error control	Avg. runtime (s)	Avg. speedup
Exact solution	–	–	–	–	6.2760	–
G-ROM	2.3×10^{-7}	5.3×10^{-8}	2.3×10^{-8}	5.4×10^{-9}	2.6526	2.37
DNN-ROM	2.2×10^{-5}	5.8×10^{-6}	9.1×10^{-6}	2.0×10^{-6}	0.1623	40.33
VKOGA-ROM	7.0×10^{-5}	1.8×10^{-5}	2.5×10^{-5}	6.9×10^{-6}	0.1580	41.03
GPR-ROM	1.4×10^{-5}	2.2×10^{-6}	4.2×10^{-6}	7.6×10^{-7}	0.1572	41.40

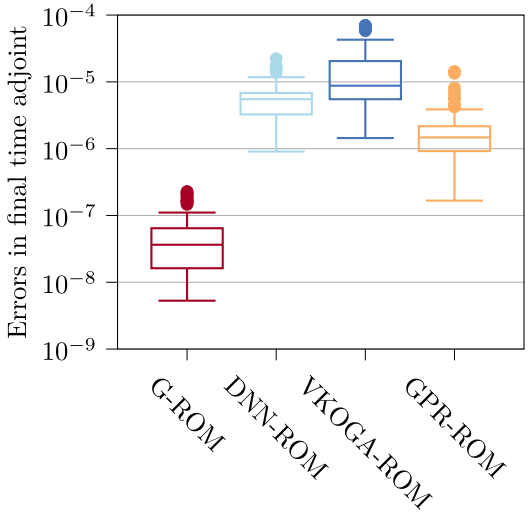

FIGURE 6. Box plot showing the statistics of the numerical results for the heat equation using the G-ROM, DNN-ROM, VKOGA-ROM and GPR-ROM.

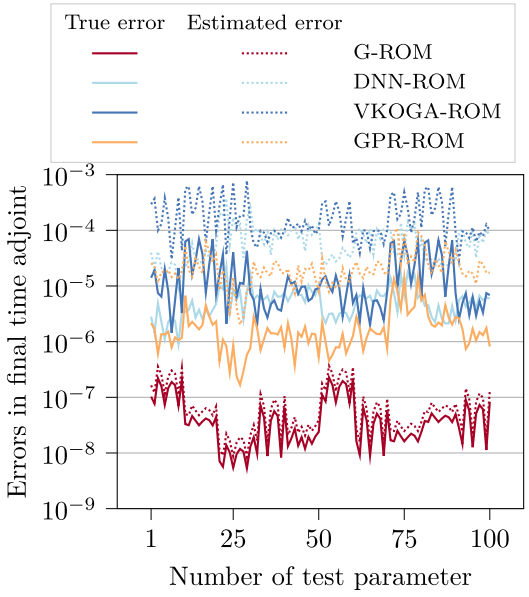

FIGURE 7. True errors and error estimation for the heat equation over the enumerated test parameters in the online phase.

the greedy algorithm and the exact optimal control problem does not have to be solved for all parameters in $\mathcal{P}_{\text{train}}$. All reduced models, the G-ROM, the DNN-ROM, the VKOGA-ROM, and the GPR-ROM have been tested on a set of 100 randomly chosen parameters that have not been part of the training set. The results are summarized in Table 1 and presented in more detail in Figures 6 and 7. Here we address two types of errors: the errors in the optimal final time adjoints, *i.e.* the deviation of $\varphi_\mu^*(T)$ from its approximate value, and similarly the errors in the optimal controls. Both, the averaged and maximal errors in the final time adjoint, are well below 10^{-4} for all considered methods. We emphasize at this point that the tolerance $\tilde{\varepsilon} = 10^{-6}$ prescribed for the greedy construction of the reduced basis is only reached by the G-ROM. The approximation of the reduced coefficients by machine learning introduces an additional error that is reflected in a larger error in the final time adjoint. Nevertheless, the machine learning algorithms provide good approximations in particular in terms of the average error. Further, the average errors in the control for all reduced models are of order 10^{-6} or even smaller and therefore the approximate controls are close to the optimal ones. The last two columns of Table 1 show average runtimes of the considered algorithms for computing the (approximate) control and speedups of the reduced models compared to the computation of the exact solution. Here we see a speedup of about 2 for the G-ROM while all machine learning surrogates reach speedups of around 40. Therefore, the machine learning

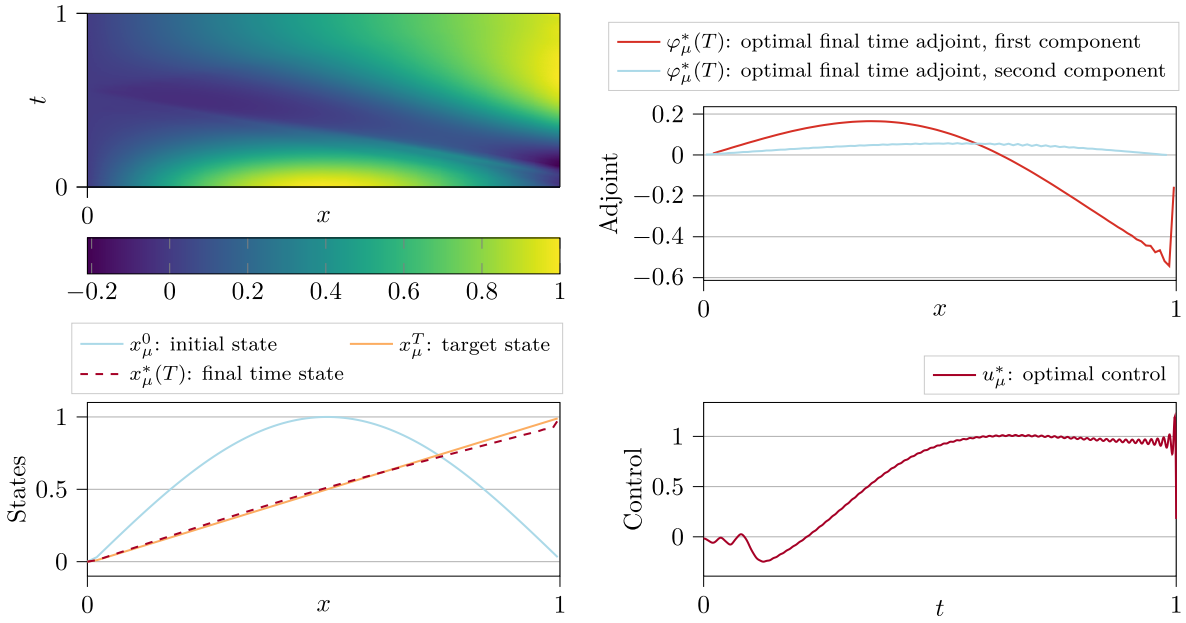

FIGURE 8. Optimal state x_μ^* in a space-time plot (*top left*), optimal final time adjoint $\varphi_\mu^*(T)$ with the first component associated to the state and second component associated to the velocity (*top right*), optimal control u_μ^* (*bottom right*) and initial x_μ^0, final $x_\mu^*(T)$ and target x_μ^T states (*bottom left*) for the parameter $\mu = 5$ in the damped wave equation example. The velocities are not shown in the plots of the states on the left-hand side, as they are all close to being zero. For the optimal final time adjoint, both components are plotted since these have to be approximated by the reduced space.

reduced models outperform the G-ROM in terms of computational efficiency during the online phase. In this example, in particular the GPR-ROM constitutes a serious alternative to the G-ROM also with respect to approximation accuracy. We omit a detailed listing of the training times for the machine learning models since they are performed completely offline and amount to a couple of seconds at most. They are negligible compared to the time required for building the reduced basis using the greedy procedure. In Figure 7 we present the true errors and the estimated errors for the various reduced models over the enumerated test parameter set. The estimated errors are about an order of magnitude larger for the machine learning models compared to the true errors. Accordingly, the efficiency constant of the error estimator η_μ seems to be relatively moderate, which results in a good error estimate for the machine learning models. It is also visible that the estimator indeed provides a reliable estimate, *i.e.* the estimated error is an upper bound for the actual error. For the G-ROM we in particular observe that both, the true and the estimated error in the final time adjoint, are below the prescribed tolerance of 10^{-6} over the whole parameter test set.

6.3. Test case 2: Damped wave equation

In our second numerical example we consider a parametrized damped wave equation with a Dirichlet control acting on the right boundary of the one-dimensional spatial domain. The problem of interest reads as

$$\begin{aligned}
\partial_{tt} v_\mu(t,y) + \nu \partial_t v_\mu(t,y) - \mu \Delta v_\mu(t,y) &= 0 && \text{for } t \in [0,T], y \in \Omega,\\
v_\mu(t,0) &= 0 && \text{for } t \in [0,T],\\
v_\mu(t,1) &= u_\mu(t) && \text{for } t \in [0,T],\\
v_\mu(0,y) &= v_\mu^0(y) = \sin(\pi y) && \text{for } y \in \Omega,\\
\partial_t v_\mu(0,y) &= 0 && \text{for } y \in \Omega.
\end{aligned}$$

The damping constant is chosen as $\nu = 10$, see the discussion below as well. Further, we choose as final time $T = 1$, the spatial domain $\Omega = [0, 1]$, while the parameter μ, determining the velocity, ranges within the interval $\mathcal{P} = [3, 10] \subset \mathbb{R}$. Similarly as in the example of the heat equation we discretize the spatial derivative by means of second-order central finite differences on a grid with $n_y = 100$ inner grid points. For time discretization we apply the Crank-Nicolson scheme with $n_t = 10 \cdot n_y$ time steps. We rewrite the discretized damped wave equation as a first order system of dimension $n = 2 \cdot n_y = 200$. The resulting system matrices are given as

$$A_\mu = \left[\begin{array}{c|c} 0 & I \\ \hline \frac{\mu}{h^2}\bar{A} & -\nu I \end{array}\right] \in \mathbb{R}^{n\times n} \qquad \text{and} \qquad B_\mu = \frac{\mu}{h^2}\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \in \mathbb{R}^{n\times 1},$$

where the matrix $\bar{A} \in \mathbb{R}^{n_y\times n_y}$ was defined above for the heat equation example in equation (29), and $I \in \mathbb{R}^{n_y\times n_y}$ denotes the identity matrix. Similar to the heat equation example, the discretized system is controllable, *i.e.* the Kalman rank condition is fulfilled. Furthermore, the parameter to system matrices mappings are Lipschitz continuous and therefore Algorithm 2.1 holds as well.

We prescribe the target $v_\mu^T(y) = y$ and $\partial_t v_\mu^T(y) = 0$ for all $y \in \Omega$ and $\mu \in \mathcal{P}$. Further, we choose the same inner products and norms as in the previous example of the heat equation, see equations (30) and (31). In this example we set $M = 10 \cdot I \in \mathbb{R}^{n\times n}$ and $R = 0.1 \cdot I \in \mathbb{R}^{1\times 1}$ for the weighting matrices associated to the different terms in the objective functional. Hence, also Assumption 2.6 is fulfilled in this case.

Figure 8 depicts the optimal final time adjoint, optimal control and corresponding optimal states (as space-time plot and at final time T) for the damped wave equation with parameter $\mu = 5 \in \mathcal{P}$. The initial state is driven towards the target state as can be seen in the plots on the left hand side of the figure. As in the heat equation example, the target state is not hit exactly but a deviation is allowed. The velocity component of the states is not shown in the plot since the target and initial velocity are zero and the velocity of the optimal state at final time is of order 10^{-3} or smaller. The top right plot shows the optimal final time adjoint divided into its two components corresponding to the state and the velocity. One can see that the final time adjoint state is quite smooth for both components and that in particular the second component, associated to the velocity, is relatively close to zero. This facilitates a low-dimensional approximation of the final time adjoint. The bottom right plot of Figure 8 shows the optimal control which exhibits the typical oscillations patterned at the end of the time interval. These oscillations are much more pronounced than the oscillations in the optimal final time adjoint. Hence, it seems advantageous to approximate the adjoint datum instead of the control.

The plots in Figure 8 depict the solution for a single parameter value $\mu = 5$. In order to deal with the full range of parameter dependent problems, we apply the greedy Algorithm 1 with a training set consisting of $n_{\text{train}} = 50$ training samples from a uniform grid on the parameter domain $\mathcal{P}$. The greedy tolerance is set to $\tilde{\varepsilon} = 10^{-2}$. Similar to the previous example we have no prior knowledge of the constants C_Λ and γ and therefore directly select $\tilde{\varepsilon}$ instead of ε.

Figure 9 shows the estimated maximum and true maximum errors over the greedy steps until the greedy tolerance $\tilde{\varepsilon}$ is reached for a reduced basis size of $N = 18$. The parameters selected by the greedy algorithm are also shown in Figure 12 (marked as crosses ×) and are distributed almost uniformly across the parameter range with a slight clustering at the smaller parameter values. One can observe in Figure 9 that the estimator is more pessimistic in this example than in the heat equation problem and overestimates the true error by about two orders of magnitude. Nevertheless, the error estimator still provides a reliable estimate of the error, *i.e.* the estimated error is an upper bound for the true error. However, the greedy procedure is not stopped with a reduced size of 5 although the true error would have been already below the prescribed tolerance for such a basis size. This behavior reveals the fact that the constant C_Λ, which determines the efficiency of the error estimator, is larger for this example. This is to be expected, as the dissipation rate is weaker in this case than it was for the heat equation example.

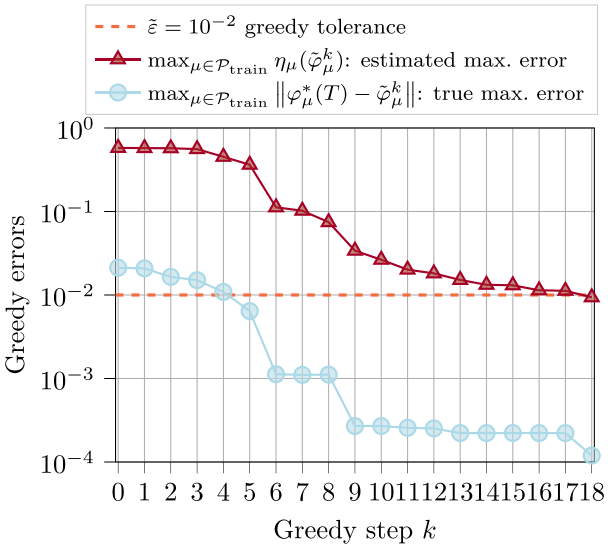

FIGURE 9. Results of the greedy algorithm applied to the damped wave equation example.

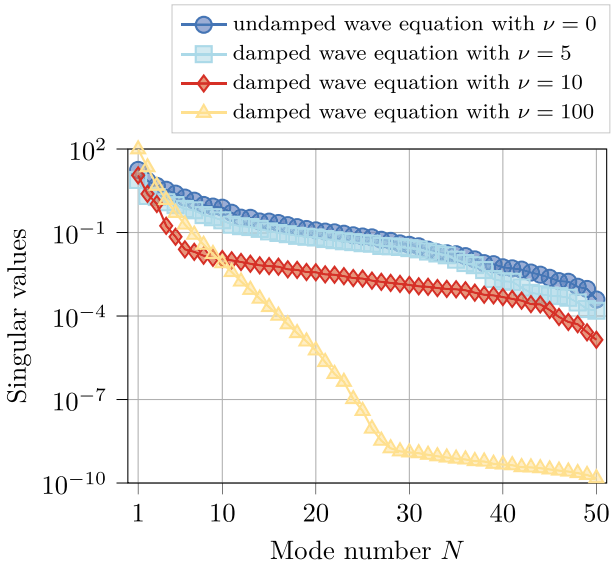

FIGURE 10. Singular value decay of optimal final time adjoints for the damped wave equation.

In order to explore how the dissipation rate influences the problem, we analyze its dependence on the damping constant ν. Figure 10 shows the singular values for the optimal final time adjoints over the training set for different values of the damping constant ν. The singular value decay is rather slow in the case of the undamped wave equation, *i.e.* for $\nu = 0$, see also [15]. The same behavior emerges for a small value of the damping constant like $\nu = 5$, which means that also the decay of the Kolmogorov N-width of the manifold of optimal final time adjoints is slow in these cases. Thus, a relatively large reduced space would be necessary to obtain a sufficiently accurate reduced model. In that regard, the restriction to linear subspaces (constructed by the greedy algorithm in this case) can be seen as a limitation of the general methodology since it might not be applicable to all control systems. The introduction of a larger damping constant such as $\nu = 10$ or even $\nu = 100$, however, leads to a faster decay of the singular values (see the curves marked as and in Fig. 10) and thus makes the solution manifold amenable for approximation by linear subspaces of small dimension. For a large damping constant of $\nu = 100$ we observe an exponential decay of the singular values (similar to the heat equation example) up to a reduced basis size of around 30. For a moderate damping constant of $\nu = 10$, the decay is also exponential at least for the first couple of modes. We should nevertheless mention that also in the case of the damped wave equation with $\nu = 10$, the decay of the singular values slows down quite rapidly after about 7 modes. Hence, to reach very small approximation errors, still a large reduced basis would be required. Since we are not interested in an example where the dissipativity, resulting from a large damping constant, dominates the solution behavior, we choose a value of $\nu = 10$ and the larger greedy tolerance of $\tilde{\varepsilon} = 10^{-2}$.

After constructing a reduced basis using the greedy procedure from Section 3.2, the machine learning reduced models described in Section 4 are trained on the same $n_{\text{train}} = 50$ training parameters that were already used in the greedy algorithm. For testing purposes, we draw 100 random values from the parameter range $\mathcal{P}$ that were not contained in the training set and evaluate the performance of the reduced models on the test parameter set. The results of the online phase are presented in Table 2. Furthermore, the error statistics for the reduced models are summarized in Figure 11, whereas the errors of the reduced models with respect to the test parameters are shown in detail in Figure 12. First of all, we observe an enormous speedup reached by all machine learning models compared to the exact solution of the optimal control and also compared to the G-ROM. The average

TABLE 2. Results of the numerical experiments for the damped wave equation using the G-ROM, DNN-ROM, VKOGA-ROM and GPR-ROM.

Method	Max. error adjoint	Avg. error adjoint	Max. error control	Avg. error control	Avg. runtime (s)	Avg. speedup
Exact solution	–	–	–	–	228.8106	–
G-ROM	3.0×10^{-4}	4.7×10^{-5}	1.3×10^{-4}	2.3×10^{-5}	10.8503	21.10
DNN-ROM	3.8×10^{-3}	3.8×10^{-4}	4.6×10^{-3}	7.0×10^{-4}	0.3230	731.48
VKOGA-ROM	2.0×10^{-2}	5.7×10^{-4}	5.7×10^{-3}	2.0×10^{-4}	0.3226	733.80
GPR-ROM	8.9×10^{-3}	3.9×10^{-4}	1.1×10^{-2}	5.3×10^{-4}	0.3359	708.55

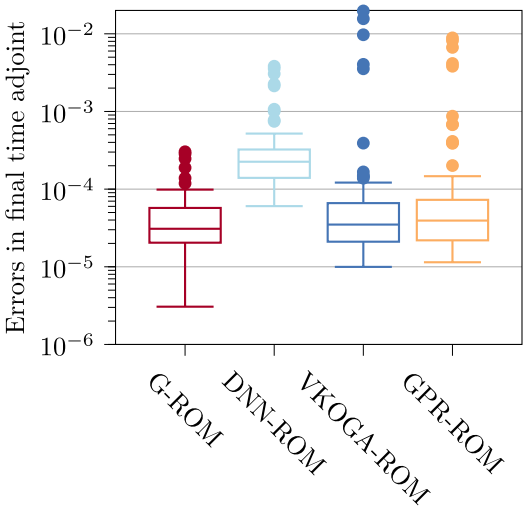

FIGURE 11. Box plot showing the statistics of the numerical results for the damped wave equation using the G-ROM, DNN-ROM, VKOGA-ROM and GPR-ROM.

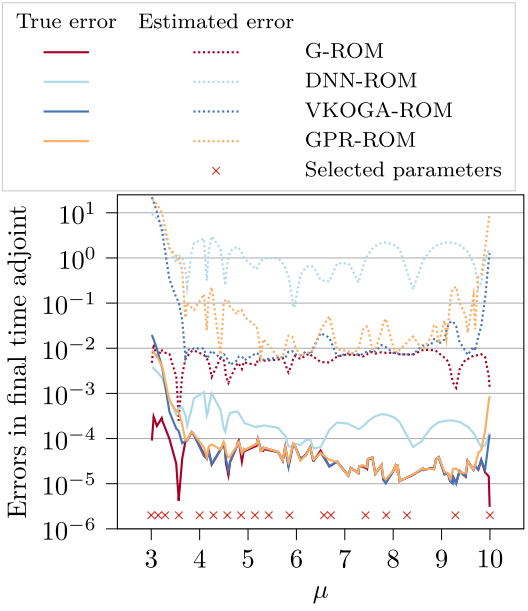

FIGURE 12. True errors and error estimation for the damped wave equation over the test parameters in the online phase. The parameter values selected during the greedy algorithm are shown as crosses.

speedup of the latter model is about 21, while the average speedup of the three considered machine learning surrogates is around 730. In total, the average runtime could be reduced from about 229 s for the exact solution to roughly 0.32 s using the combined machine learning/reduced basis surrogates. In terms of approximation accuracy, we see that the average error in the final time adjoint is about one order of magnitude larger for the machine learning reduced models compared to the G-ROM. For the control we observe that the DNN-ROM, the VKOGA-ROM, and the GPR-ROM all reach average errors of the order 10^{-4}. The box plot in Figure 11 and also the true errors in Figure 12 reveal that the VKOGA-ROM and the GPR-ROM actually suffer from several outliers, in particular at the boundaries of the parameter domain, which limit their average behavior. However, on most parts of the parameter domain the error of those two methods is close to the error of the G-ROM. The DNN-ROM on the other hand produces errors that are about one order of magnitude larger on the whole parameter domain but with less outliers. It might be possible to reduce these outliers and further improve the performance of the machine learning models by tuning their hyper-parameters. However, such an optimization of the machine learning models is beyond the scope of this article and not the focus of this work. To adaptively

adjust for the outliers, we remark that the error estimator also captures the outliers well and might therefore help to detect areas in the parameter domain where additional training data should be generated to improve the machine learning models. At this point, we also would like to advert to the relatively large overestimation of the error that can be observed in Figure 12 and was already present in Figure 9. Also for the most accurate reduced model, *i.e.* the G-ROM, the estimated error is about two orders of magnitude larger than the true error. Nevertheless, the curves of estimated errors follow the true errors quite closely in terms of error variation with respect to the parameter, although being two orders of magnitude larger.

7. Conclusion and outlook

In this contribution we considered parameterized optimal control problems where the objective functional penalizes a (weighted) deviation from a target state and a large control energy. To solve such optimal control problems fast for several values of the parameter, for instance in a real-time application or many-query scenario, we first extended the greedy control procedure, previously developed in the controllability context, either the exact [29], or the approximate one [30], to this setting. In contrast to the penalisation approach applied in this work, the desired target state is considered as a constraint in the cited papers. However, both approaches allow for an application of the Hilbert uniqueness method. In particular, the optimal control, *i.e.* the one that minimizes the objective functional, can be completely described by the optimal final state of the adjoint problem. For this reason, the greedy algorithm builds a reduced basis for the manifold of optimal final time adjoint states over the parameter set, and not for the manifold of optimal controls which is usually more complex and difficult to describe.

Secondly, we applied machine learning approaches, such as deep neural networks, kernel methods, and Gaussian process regression, to learn the reduced coefficients as a function of the parameter. We derived error bounds for the greedy approximation and proved that the proposed algorithm is indeed a weak greedy algorithm. Furthermore, we also showed how to bound the error of the machine learning with respect to the greedy error and the error in approximating the coefficients. A comparison of the computational costs reveals the enormous potential in reducing the computational effort by applying machine learning in our scenario. The numerical experiments exhibit that the machine learning surrogates can accurately approximate the reduced coefficients while providing a massive speedup compared to the exact solution of the optimal control problem. In addition, due to *a posteriori* error estimates we derived, it is possible to obtain a reliable bound on the error of the machine learning results in a cheap manner without computing the exact solution.

The numerical experiments in this paper are run for control systems whose underlying dynamics (governed by the heat and the damped wave operator) are dissipative, which is crucial for the efficiency of the greedy algorithm. Namely, a conservative system supports large, non-linear variations of optimal final time adjoints with respect to the parameter [15], which requires a relatively big reduced basis to obtain a sufficiently accurate reduced model (*cf.* the discussion on the damping constant in Sect. 6.3). In this way, the developed methods can be similarly (and successfully) applied to other dissipative systems, such as advection-diffusion-reaction ones. Although our examples are based on PDEs in one space dimension, the same procedure can be applied in a higher dimensions cases as well. Here we do not expect difficulties, apart from those conditioned by the efficiency of standard numerical methods that have to be employed in the offline phase.

As discussed extensively in Section 4, many different machine learning algorithms can be integrate in the approach. The only requirement on the machine learning surrogate is that it approximates vector-valued functions and can be trained in a supervised manner, *i.e.* by providing samples of the function to approximate. A theoretical investigation of the machine learning models in terms of approximation quality of the parameter to solution map could be a topic of future research. For instance for the kernel methods applied above, theoretical results on the approximation properties and bounds for the errors exist (*e.g.* [43]) that allow for a rigorous *a priori* error bound that involves only the P-greedy tolerance ε_P and the norm of the parameter to solution map in the reproducing kernel Hilbert space (under the assumption that this mapping is indeed contained in the reproducing kernel Hilbert space).

To further speedup the computations of the reduced models during the online phase (in particular Lines 1, 5, and 6 in Algorithm 3 and Lines 3 and 4 in Algorithm 5), an additional hyper-reduction, that accelerates solving the equation for the time-dependent adjoint variable, may be applied. Similarly, the error estimator could be approximated by replacing the system in equation (4a) by a reduced system. To guarantee a reliable and efficient error estimator, theoretical investigations of such an approach would be indispensable. However, we should mention that replacing the optimality system in equation (4) by a surrogate would also speedup the exact computation of the optimal final time adjoint. Several approaches considering parametrized model order reduction of control systems have been suggested in the last decades, see [3] for a survey of methods.

An application of the adaptive model hierarchy from [18] as described in Theorem 4.4 to the parametrized optimal control setting could be an interesting extension and combination of the approaches. The error estimators and reduced models developed in this paper would serve as the main ingredients to an adaptive model hierarchy with certified results for parametrized optimal control problems.

Finally, it is of interest to investigate extensions of the approaches discussed in this paper to other classes of optimal control problems. As an example, one might explore a generalization of the method to linear time-variant systems with time-dependent parameters, objective functionals that are not necessarily quadratic with respect to the state and the control, or to problems with additional constraints such as bounds on the control variables. Instead of a target state, it may also be of interest to include an output quantity and an objective functional measuring a deviation from a target output. Similarly, one could penalise a deviation of the whole trajectory from some desired one. This would require the study of the coupled optimality system, and the optimal feedback controls obtained by solving the corresponding Riccati equations should be explored. In such a setting, the greedy approach should probably be accompanied by POD, which is common when constructing a reduced basis for a manifold of time-dependent functions [22].

The presented theory is developed for bounded linear operators. As such, it cannot be applied directly to PDEs. Instead, in the numerical examples we apply our algorithms to finite-dimensional approximations of the original differential operators. Of course, convergence of the procedure and the results as the approximation dimension goes to infinity is completely open. Extending the theory to unbounded dynamic operators requires rewriting the optimality conditions in the PDE context, see for instance [23]. Certainly, it is an interesting direction to discover in future research.

Acknowledgements

C. Molinari is part of the Indam group “Gruppo Nazionale per l’Analisi Matematica, la Probabilità le loro applicazioni” and has been supported by the MIUR Excellence Department Project awarded to Dipartimento di Matematica, Università di Genova, CUP D33C23001110001. He acknowledges the support of the AFOSR project FA8655-22-1-7034, the support of MIUR-PRIN 202244A7YL project “Gradient Flows and Non-Smooth Geometric Structures with Applications to Optimization and Machine Learning”, the support of the European Research Council (grant SLING 819789), the support of Ministry of Education, University and Research (grant BAC FAIR PE00000013 funded by the EU - NGEU), and the support of the Programma Operativo Nazionale (PON) “Ricerca e Innovazione” 2014–2020.

Data availability statement

The source code used to carry out the numerical experiments presented in this contribution can be found in [26] and is also available in a GitHub repository under `https://github.com/HenKlei/ML-OPT-CONTROL`.

References

[1] N. Aronszajn, Theory of reproducing kernels. *Trans. Am. Math. Soc.* **68** (1950) 337–404.

[2] F. Ballarin, G. Rozza and M. Strazzullo, Chapter 9 – Space-time POD-Galerkin approach for parametric flow control, in Numerical Control: Part A. Vol. 23 of *Handbook of Numerical Analysis*, edited by E. Trélat and E. Zuazua. Elsevier, Amsterdam, The Netherlands (2022) 307–338.

[3] P. Benner, S. Gugercin and K. Willcox, A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Rev.* **57** (2015) 483–531.
[4] B. Bohn, C. Rieger and M. Griebel, A representer theorem for deep kernel learning. *J. Mach. Learn. Res.* **20** (2019) 2302–2333.
[5] A. Cohen and R. DeVore, Approximation of high-dimensional parametric PDEs. *Acta Numer.* **24** (2015) 1–159.
[6] A. Cohen and R. DeVore, Kolmogorov widths under holomorphic mappings. *IMA J. Numer. Anal.* **36** (2016) 1–12.
[7] J. Crank and P. Nicolson, A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Math. Proc. Camb. Philos. Soc.* **43** (1947) 50–67.
[8] N. Dal Santo, S. Deparis and L. Pegolotti, Data driven approximation of parametrized PDEs by reduced basis and neural networks. *J. Comput. Phys.* **416** (2020) 109550.
[9] T. Daniel, F. Casenave, N. Akkari and D. Ryckelynck, Model order reduction assisted by deep neural networks (ROM-net). *Adv. Model. Simul. Eng. Sci.* **7** (2020) 16.
[10] L. Dedè, Reduced basis method and error estimation for parametrized optimal control problems with control constraints. *SIAM J. Sci. Comput.* **50** (2012) 287–305.
[11] R. DeVore, G. Petrova and P. Wojtaszczyk, Greedy algorithms for reduced bases in Banach spaces. *Constr. Approx.* **37** (2013) 455–466.
[12] T. Ehring and B. Haasdonk, Hermite kernel surrogates for the value function of high-dimensional nonlinear optimal control problems. *Adv. Comput. Math.* **50** (2024) 36.
[13] G.H. Golub and C.F. Van Loan, Matrix Computations, 3rd edition. The Johns Hopkins University Press, Baltimore, Maryland (1996).
[14] C. Gräßle, M. Hinze and S. Volkwein, Model order reduction by proper orthogonal decomposition, in Model Order Reduction. Vol. 2 *Snapshot-Based Methods and Algorithms*, edited by P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders and L.M. Silveira. De Gruyter, Berlin, Boston (2021) 47–96.
[15] C. Greif and K. Urban, Decay of the Kolmogorov N-width for wave problems. *Appl. Math. Lett.* **96** (2019) 216–222.
[16] M. Guo and J.S. Hesthaven, Reduced order modeling for nonlinear structural analysis using Gaussian process regression. *Comput. Methods Appl. Mech. Eng.* **341** (2018) 807–826.
[17] B. Haasdonk and M. Ohlberger, Efficient reduced models and a posteriori error estimation for parametrized dynamical systems by offline/online decomposition. *Math. Comput. Model. Dyn. Syst.* **17** (2011) 145–161.
[18] B. Haasdonk, H. Kleikamp, M. Ohlberger, F. Schindler and T. Wenzel, A new certified hierarchical and adaptive RB-ML-ROM surrogate model for parametrized PDEs. *SIAM J. Sci. Comput.* **45** (2023) A1039–A1065.
[19] J.K. Hale, Ordinary Differential Equations, 2nd edition. Robert E. Krieger Publishing Company, Malabar, Florida (1980).
[20] C.R. Harris, K. Jarrod Millman, S.J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N.J. Smith, R. Kern, M. Picus, S. Hoyer, M.H. van Kerkwijk, M. Brett, A. Haldane, J.F. del Río, Mark Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke and T.E. Oliphant, Array programming with NumPy. *Nature* **585** (2020) 357–362.
[21] J.S. Hesthaven and S. Ubbiali, Non-intrusive reduced order modeling of nonlinear problems using neural networks. *J. Comput. Phys.* **363** (2018) 55–78.
[22] J.S. Hesthaven, G. Rozza and B. Stamm, Certified Reduced Basis Methods for Parametrized Partial Differential Equations. SpringerBriefs in Mathematics, Springer, Cham, New York (2016).
[23] M. Hinze, R. Pinnau, M. Ulbrich and S. Ulbrich. Optimization with PDE Constraints. Springer, Netherlands (2009).
[24] M. Kanagawa, P. Hennig, D. Sejdinovic and B.K. Sriperumbudur, Gaussian processes and kernel methods: a review on connections and equivalences (2018). Preprint: arXiv: 1807.02582.
[25] T. Keil, H. Kleikamp, R.J. Lorentzen, M.B. Oguntola and M. Ohlberger, Adaptive machine learning-based surrogate modeling to accelerate PDE-constrained optimization in enhanced oil recovery. *Adv. Comput. Math.* **48** (2022) 73.
[26] H. Kleikamp, M. Lazar and C. Molinari, Python code for "Be greedy and learn: Efficient and certified algorithms for parametrized optimal control problems" (2023). https://zenodo.org/records/8188417.
[27] T. Kmet, Neural network solution of optimal control problem with control and state constraints in Artificial Neural Networks and Machine Learning – ICANN 2011, edited by T. Honkela, W. Duch, M. Girolami and S. Kaski. Springer, Berlin, Heidelberg (2011) 261–268.
[28] M. Lazar and J. Lohéac, Chapter 8 – Control of parameter dependent systems, in *Numerical Control: Part A*, Vol. 23 of *Handbook of Numerical Analysis*, edited by E. Trélat and E. Zuazua. Elsevier, Amsterdam (2022) 265–306.
[29] M. Lazar and E. Zuazua, Greedy controllability of finite dimensional linear systems. *Automatica* **74** (2016) 327–340.
[30] M. Lazar and E. Zuazua, Greedy search of optimal approximate solutions. *Pure Appl. Funct. Anal.* **8** (2023) 547–564.

[31] Y. LeCun, Y. Bengio and G. Hinton, Deep learning. *Nature* **521** (2015) 436–444.

[32] D.C. Liu and J. Nocedal, On the limited memory BFGS method for large scale optimization. *Math. Program.* **45** (1989) 503–528.

[33] J. Mayer, M. Dolgov, T. Stickling, S. Özgen, F. Rosenthal and U.D. Hanebeck, Stochastic optimal control using Gaussian process regression over probability distributions, in 2019 American Control Conference (ACC). Philadelphia (2019) 4847–4853.

[34] R. Milk, S. Rave and F. Schindler, pyMOR – Generic algorithms and interfaces for model order reduction. *SIAM J. Sci. Comput.* **38** (2016) S194–S216.

[35] C. Molinari, M. Massias, L. Rosasco and S. Villa, Iterative regularization for convex regularizers, in International Conference on Artificial Intelligence and Statistics. Vol. 130 of *Proceedings of Machine Learning Research*, edited by A. Banerjee and K. Fukumizu. PMLR, San Diego, California (2021) 1684–1692.

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala, Pytorch: an imperative style, high-performance deep learning library. In: *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox and R. Garnett. Curran Associates, Inc., New York (2019) 8024–8035.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and É. Duchesnay, Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12** (2011) 2825–2830.

[38] P. Petersen and F. Voigtlaender, Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Netw.* **108** (2018) 296–330.

[39] J. Peypouquet, Convex Optimization in Normed Spaces: Theory, Methods and Examples. Springer International Publishing, New York (2015).

[40] L. Prechelt, Early stopping – but when? In: *Neural Networks: Tricks of the Trade, volume 1524 of LNCS, chapter 2*. Springer, Berlin, Heidelberg (1997) 55–69.

[41] C.E. Rasmussen and C.K.I. Williams, Gaussian Processes for Machine Learning. Adaptive Computation and Machine Learning. MIT Press, Massachusetts (2006).

[42] D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning representations by back-propagating errors. *Nature* **323** (1986) 533–536.

[43] G. Santin and B. Haasdonk, Convergence rate of the data-independent P-greedy algorithm in kernel-based approximation. *Dolomit. Res. Notes Approx.* **10** (2017) 68–78.

[44] G. Santin and B. Haasdonk, Kernel methods for surrogate modeling. In Vol. 2 *Model Order Reduction*, edited by P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders and L.M. Silveira. De Gruyter, Berlin, Boston (2021).

[45] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S.J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A.R.J. Nelson, E. Jones, R. Kern, E. Larson, C.J. Carey, İ. Polat, Y.Feng, E.W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E.A. Quintero, C.R. Harris, A.M. Archibald, A.H. Ribeiro, F. Pedregosa, P. van Mulbregt and SciPy 1.0 Contributors, SciPy 1.0: fundamental algorithms for scientific computing in python. *Nat. Methods* **17** (2020) 261–272.

[46] Q. Wang, J.S. Hesthaven and D. Ray, Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem. *J. Comput. Phys.* **384** (2019) 289–307.

[47] H. Wendland, Scattered data approximation. In Vol. 17 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge (2005).

[48] T. Wenzel, G. Santin and B. Haasdonk, A novel class of stabilized greedy kernel approximation algorithms: Convergence, stability and uniform point distribution. *J. Approx. Theory* **262** (2021) 105508.

[49] T. Wenzel, B. Haasdonk, H. Kleikamp, M. Ohlberger and F. Schindler, Application of deep kernel models for certified and adaptive RB-ML-ROM surrogate modeling. In: *Lecture Notes in Computer Science*. Springer, Nature Switzerland (2024) 117–125.

Appendix A. Proof of Theorem 2.4

For simplicity, we omit the dependence of the involved quantities on the parameter $\mu \in \mathcal{P}$.

Proof. Let $u^* \in G$ denote an optimal control, $x^* \in H$ the corresponding state trajectory. We are going to prove that the first variation of $\mathcal{J}$ vanishes if x^*, φ^* and u^* solve the boundary value problem stated above. To this end, let $v \in G$, and consider the perturbation $u \in G$ of u^* defined as

$$u(t) \coloneqq u^*(t) + \varepsilon v(t)$$

for $\varepsilon \in \mathbb{R}$. Hence, the state equation for the control u reads

$$\dot{x}(t) = Ax(t) + Bu^*(t) + \varepsilon Bv(t) \qquad \text{for } t \in [0,T].$$

The solution is explicitly given by

$$\begin{aligned} x(t) &= e^{At}x^0 + \int_0^t e^{A(t-s)}\big(Bu^*(s) + \varepsilon Bv(s)\big)\,\mathrm{d}s \\ &= x^*(t) + \varepsilon \int_0^t e^{A(t-s)}Bv(s)\,\mathrm{d}s \\ &= x^*(t) + \varepsilon z(t) \end{aligned}$$

for $z \in H$ defined as $z(t) \coloneqq \int_0^t e^{A(t-s)}Bv(s)\,\mathrm{d}s$. It holds that z satisfies the differential equation

$$\dot{z}(t) = Az(t) + Bv(t), \qquad z(0) = 0.$$

Introducing the adjoint state $\varphi \in H$ and the Hamiltonian function $\mathcal{H}\colon X \times U \times X \to \mathbb{R}$ given as

$$\mathcal{H}(x(t), u(t), \varphi(t)) = \frac{1}{2}\langle u(t), Ru(t)\rangle + \langle \varphi(t), \big(Ax(t) + Bu(t)\big)\rangle,$$

we can rewrite the functional $\mathcal{J}$ as

$$\mathcal{J}(u) = \frac{1}{2}\langle x(T) - x^T, M\left(x(T) - x^T\right)\rangle + \int_0^T \mathcal{H}(x(t), u(t), \varphi(t)) - \langle \varphi(t), \dot{x}(t)\rangle\,\mathrm{d}t,$$

since $x \in H$ solves the state equation. This holds for any adjoint state $\varphi \in H$. Similarly, for the optimal control u^* and corresponding state trajectory x^* it holds

$$\mathcal{J}(u^*) = \frac{1}{2}\langle x^*(T) - x^T, M\left(x^*(T) - x^T\right)\rangle + \int_0^T \mathcal{H}(x^*(t), u^*(t), \varphi(t)) - \langle \varphi(t), \dot{x}^*(t)\rangle\,\mathrm{d}t.$$

We now consider the difference $\mathcal{J}(u) - \mathcal{J}(u^*)$, which is given as

$$\begin{aligned}\mathcal{J}(u) - \mathcal{J}(u^*) = \frac{1}{2}\left[\langle x(T) - x^T, M\left(x(T) - x^T\right)\rangle - \langle x^*(T) - x^T, M\left(x^*(T) - x^T\right)\rangle\right] \\ + \int_0^T \mathcal{H}(x(t), u(t), \varphi(t)) - \mathcal{H}(x^*(t), u^*(t), \varphi(t))\,\mathrm{d}t \\ + \int_0^T \langle \varphi(t), \dot{x}^*(t) - \dot{x}(t)\rangle\,\mathrm{d}t.\end{aligned} \tag{A.1}$$

We obtain for the first term in (A.1) the identity

$$\begin{aligned}&\frac{1}{2}\left[\langle x(T) - x^T, M\big(x(T) - x^T\big)\rangle - \langle x^*(T) - x^T, M\big(x^*(T) - x^T\big)\rangle\right] \\ &\quad = \frac{1}{2}\left[\langle x^*(T) + \varepsilon z(T) - x^T, M\big(x^*(T) + \varepsilon z(T) - x^T\big)\rangle - \langle x^*(T) - x^T, M\big(x^*(T) - x^T\big)\rangle\right] \\ &\quad = \varepsilon\langle z(T), M\big(x^*(T) - x^T\big)\rangle + \mathcal{O}(\varepsilon^2),\end{aligned}$$

where we used that M is self-adjoint. For the difference of the Hamiltonians in the second term in (A.1) it holds

$$\begin{aligned}&\mathcal{H}(x(t), u(t), \varphi(t)) - \mathcal{H}(x^*(t), u^*(t), \varphi(t)) \\ &\quad = \frac{1}{2}\langle u(t), Ru(t)\rangle + \langle \varphi(t), Ax(t) + Bu(t)\rangle - \frac{1}{2}\langle u^*(t), Ru^*(t)\rangle - \langle \varphi(t), Ax^*(t) + Bu^*(t)\rangle \\ &\quad = \frac{1}{2}\langle u^*(t) + \varepsilon v(t), R\big(u^*(t) + \varepsilon v(t)\big)\rangle + \langle \varphi(t), Ax^*(t) + \varepsilon Az(t) + Bu^*(t) + \varepsilon Bv(t)\rangle \\ &\qquad - \frac{1}{2}\langle u^*(t), Ru^*(t)\rangle - \langle \varphi(t), Ax^*(t) + Bu^*(t)\rangle \\ &\quad = \varepsilon\langle u^*(t), Rv(t)\rangle + \langle \varphi(t), \varepsilon Az(t) + \varepsilon Bv(t)\rangle + \mathcal{O}(\varepsilon^2) \\ &\quad = \varepsilon\Big[\langle u^*(t), Rv(t)\rangle + \langle \varphi(t), Az(t)\rangle + \langle \varphi(t), Bv(t)\rangle\Big] + \mathcal{O}(\varepsilon^2) \\ &\quad = \varepsilon\Big[\langle Ru^*(t) + B^*\varphi(t), v(t)\rangle + \langle \varphi(t), Az(t)\rangle\Big] + \mathcal{O}(\varepsilon^2)\end{aligned}$$

for all $t \in [0, T]$, where we used that R is a self-adjoint operator. Further, recall that it holds $x(t) = x^*(t) + \varepsilon z(t)$ and therefore $\dot{x}^*(t) - \dot{x}(t) = -\varepsilon\dot{z}(t)$ for all $t \in [0, T]$. For the last term in (A.1) we thus obtain *via* integration by parts

$$\begin{aligned}\int_0^T \langle \varphi(t), \dot{x}^*(t) - \dot{x}(t)\rangle\,\mathrm{d}t &= -\varepsilon\int_0^T \langle \varphi(t), \dot{z}(t)\rangle\,\mathrm{d}t \\ &= \big[-\varepsilon\langle \varphi(t), z(t)\rangle\big]_0^T + \varepsilon\int_0^T \langle \dot{\varphi}(t), z(t)\rangle\,\mathrm{d}t \\ &= -\varepsilon\langle \varphi(T), z(T)\rangle + \varepsilon\int_0^T \langle \dot{\varphi}(t), z(t)\rangle\,\mathrm{d}t,\end{aligned}$$

where we used that it holds $z(0) = 0$. Bringing everything together gives

$$\begin{aligned}\mathcal{J}(u) - \mathcal{J}(u^*) &= \varepsilon \left[\langle z(T), M\big(x^*(T) - x^T\big)\rangle + \int_0^T \langle Ru^*(t) + B^*\varphi(t), v(t)\rangle + \langle \varphi(t), Az(t)\rangle \,\mathrm{d}t \right. \\ &\qquad \left. + \int_0^T \langle \dot{\varphi}(t), z(t)\rangle \,\mathrm{d}t - \langle \varphi(T), z(T)\rangle \right] + \mathcal{O}(\varepsilon^2) \\ &= \varepsilon \left[\langle z(T), M(x^*(T) - x^T - \varphi(T))\rangle + \int_0^T \langle Ru^*(t) + B^*\varphi(t), v(t)\rangle \,\mathrm{d}t \right. \\ &\qquad \left. + \int_0^T \langle \dot{\varphi}(t) + A^*\varphi(t), z(t)\rangle \,\mathrm{d}t \right] + \mathcal{O}(\varepsilon^2).\end{aligned}$$

Since u^* is assumed to be an optimal control, it has to hold for all $v \in U$ that

$$\begin{aligned}0 &= \lim_{\varepsilon \to 0} \frac{\mathcal{J}(u^* + \varepsilon v) - \mathcal{J}(u^*)}{\varepsilon} \\ &= \lim_{\varepsilon \to 0} \frac{\mathcal{J}(u) - \mathcal{J}(u^*)}{\varepsilon} \\ &= \langle z(T), M(x^*(T) - x^T) - \varphi(T)\rangle \\ &\qquad + \int_0^T \langle Ru^*(t) + B^*\varphi(t), v(t)\rangle \,\mathrm{d}t + \int_0^T \langle \dot{\varphi}(t) + A^*\varphi(t), z(t)\rangle \,\mathrm{d}t.\end{aligned}$$

This yields the claimed necessary conditions for u^*, x^* and φ^*. □