Zero-Shot Transferable Solution Method for Parametric Optimal Control Problems

Xinjian Li, Kelvin Kan, Deepanshu Verma, Krishna Kumar, Stanley Osher and Ján Drgoňa

***Abstract*—This paper presents a transferable solution method for optimal control problems with varying objectives using function encoder (FE) policies. Traditional optimization-based approaches must be re-solved whenever objectives change, resulting in prohibitive computational costs for applications requiring frequent evaluation and adaptation. The proposed method learns a reusable set of neural basis functions that spans the control policy space, enabling efficient zero-shot adaptation to new tasks through either projection from data or direct mapping from problem specifications. The key idea is an offline–online decomposition: basis functions are learned once during offline imitation learning, while online adaptation requires only lightweight coefficient estimation. Numerical experiments across diverse dynamics, dimensions, and cost structures show our method delivers near-optimal performance with minimal overhead when generalizing across tasks, enabling semi-global feedback policies suitable for real-time deployment.**

I. INTRODUCTION

Optimal control problems arise ubiquitously across engineering disciplines [5], [11], [40], [34]. Although the fundamental mathematical framework remains consistent, practical applications often require solving parametric problems where objectives vary according to task specifications, such as target locations in trajectory planning, terrain characteristics in mobile robotics, or process requirements in manufacturing. Classical local solution methods [39] are relatively fast but must be solved anew for each instance, whereas global solution methods based on the Hamilton–Jacobi–Bellman equations [16] are intractable in high dimensions. Machine learning-based approaches aim to bridge this gap and have achieved considerable success [42], [31]; however, they are typically tied to a fixed objective and lack transferability across tasks.

This paper addresses the challenge of efficiently adapting control policies to new objectives without solving each problem instance from scratch. The key insight is to approximate the function space of control policies using a function encoder (FE) [23], which learns a reusable set of neural network–parameterized functions. Policies for new tasks are then expressed as linear combinations of these basis functions, with task-specific coefficients inferred in a zero-shot manner either from limited trajectory data (LS) or directly from the problem specification (operator). As such, the computation cost and data requirement for online adaptation are greatly reduced, enabling efficient transfer without sacrificing accuracy.

XL and K. Kumar are with Oden Institute, UT Austin; K. Kan and SO are with Department of Mathematics, UCLA; DV is with Department of Mathematical and Statistical Sciences, Clemson University; JD is with Department of Civil and Systems Engineering, Johns Hopkins University.

This work was partially funded by NSF 2339678 and 2321040, DOE under award B&R# KJ0401010 , FWP# CC147, and ROSEI at JHU.

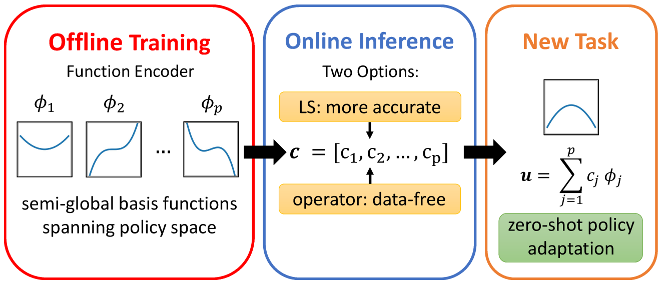

Offline Training
Function Encoder
ϕ_1
ϕ_2
ϕ_p
semi-global basis functions spanning policy space
Online Inference
Two Options:
LS: more accurate
$\boldsymbol{c} = [c_1, c_2, \dots, c_p]$
operator: data-free
New Task
$\boldsymbol{u} = \sum_{j=1}^{p} c_j \phi_j$
zero-shot policy adaptation

Fig. 1: Function encoder policies: the online–offline decoupling enables efficient and accurate policy adaptation to different optimal control objectives.

The main contributions of the work are:

- An imitation learning-based framework for parametric optimal control problems that allows for zero-shot generalization to unseen problem instances without model retraining.
- A semi-global feedback formulation that works for arbitrary inputs and is particularly well-suited when repeated evaluation of the model is required.
- Validation through extensive numerical experiments, demonstrating the robustness and near-optimal accuracy on high-dimensional and nonlinear examples.

II. BACKGROUND

This section details the necessary background, covering optimal control and the function encoder framework.

Parametric Optimal Control Problem

Given an initial state $\mathbf{x}(0) = \mathbf{x}_0$, we consider a class of optimal control problems where the system dynamics have a fixed form but the objective functional varies with each new task specification. The system evolves according to

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(0) = \mathbf{x}_0, \tag{1}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the state, $\mathbf{u}(t) \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input, and $f : \mathbb{R}^n \times \mathbb{R}^m \times [0, T] \to \mathbb{R}^n$ is the dynamics function. For each task, defined by a conditional variable $\boldsymbol{\eta}$, the objective functional in parametric form is

$$\min_{\mathbf{u}\in\mathcal{U}} \mathcal{J}(\mathbf{u}; \boldsymbol{\eta}) = \int_0^T L(\mathbf{x}(t), \mathbf{u}(t), t; \boldsymbol{\eta})\, dt + G(\mathbf{x}(T); \boldsymbol{\eta}), \tag{2}$$

where $L : \mathbb{R}^n \times \mathbb{R}^m \times [0,T] \to \mathbb{R}$ is the running cost, $G : \mathbb{R}^n \to \mathbb{R}$ is the terminal cost, and $\mathcal{U}$ is the space of admissible control functions, which we assume to be sufficiently regular [16, Sec. I.3, I.8-9]. For simplicity and without loss of generality, we use $\boldsymbol{\eta}$ to denote the conditional variable that captures task-specific variations in $\mathcal{J}$, such as target locations, terrain types, or changes in control penalties. In practice, $\boldsymbol{\eta}$ can be complex, high-dimensional or implicit. Each choice of $\boldsymbol{\eta}$ defines a distinct optimal control problem.

Given the open-loop definition in (2), lets consider a closed-loop feedback form [30] given as

$$\mathbf{u}(\mathbf{x}(t), t; \boldsymbol{\eta}) : \mathbb{R}^n \times [0,T] \to \mathcal{U} \subseteq \mathbb{R}^m, \tag{3}$$

for each specific task. Having the control $\mathbf{u}$ to depend on both the current state $\mathbf{x}(t)$ and time t allows for control evaluation for any state-time pair, making it more flexible and particularly useful in problems where the initial state $\mathbf{x}_0$ can vary. Classical results show that under suitable regularity assumptions, existence of optimal controls in feedback form can be guaranteed; see, e.g., [16, Sec. I.5], or [6].

Function Encoder

Function encoder (FE) [23] provides a principled framework for representing and transferring tasks in Hilbert spaces by learning a finite set of neural network basis functions. Given a Hilbert space $\mathcal{H}$, the FE learns basis functions $\{\phi_1, \phi_2, \dots, \phi_p\}$, parameterized by neural networks, such that any function $f \in \mathcal{H}$ can be approximated as

$$f(\mathbf{x}) \approx \sum_{j=1}^{p} c_j \phi_j(\mathbf{x}; \boldsymbol{\theta}_j), \tag{4}$$

for some $\mathbf{c} := [c_1, c_2, \dots, c_p]^\top \in \mathbb{R}^p$. Here we use $\boldsymbol{\theta} := \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_p\}$ to denote the trainable parameters of the basis functions. This linear representation enables inductive transfer of tasks: new functions are represented by inferring the coefficients $\mathbf{c}$ directly from task-specific measurements without retraining the basis functions.

Online computation of coefficients for a given new problem f is obtained by solving a least-squares (LS) problem

$$\mathbf{c} = \underset{\mathbf{c}\in\mathbb{R}^p}{\arg\min} \, \| f - \sum_{j=1}^{p} c_j \phi_j \|_{\mathcal{H}}^2. \tag{5}$$

This admits a closed-form solution given by

$$\mathbf{c} = \begin{bmatrix} \langle \phi_1, \phi_1 \rangle_{\mathcal{H}} & \cdots & \langle \phi_1, \phi_p \rangle_{\mathcal{H}} \\ \vdots & \ddots & \vdots \\ \langle \phi_p, \phi_1 \rangle_{\mathcal{H}} & \cdots & \langle \phi_p, \phi_p \rangle_{\mathcal{H}} \end{bmatrix}^{-1} \begin{bmatrix} \langle f, \phi_1 \rangle_{\mathcal{H}} \\ \vdots \\ \langle f, \phi_p \rangle_{\mathcal{H}} \end{bmatrix}, \tag{6}$$

where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the inner product associated with the Hilbert space $\mathcal{H}$, a common choice is the L^2 inner product. Both the matrix, known as the Gram matrix, and the right-hand side vector of (6) can be computed directly from task-related measurements or data, using Monte Carlo integration. In practice, we also incorporate Tikhonov regularization to (5) to improve numerical stability; see [17]. Importantly, FE are supported by a strong theoretical guarantee.

Theorem 1 (Universal Function Space Approximation [23]). *Let $K \subset \mathbb{R}^n$ be compact and let $\mathcal{H} = \{f : K \to \mathbb{R}^n \mid \|f\|_{\mathcal{H}} < \infty\}$ be a separable Hilbert space. For any continuous $f \in \mathcal{H}$ and any $\epsilon > 0$, there exist neural network basis functions $\{\phi_j\}_{j=1}^P$ and coefficients $\{c_j\}_{j=1}^P$ such that*

$$\Big\| f - \sum_{j=1}^{P} c_j \, \phi_j \Big\|_{\mathcal{H}} < \epsilon \, \|f\|_{\mathcal{H}}.$$

Here Theorem 1 establishes that, with a sufficient number of basis functions, FEs can approximate any function in $\mathcal{H}$ with arbitrary precision, making them a principled and general-purpose tool for our transfer learning task at hand.

III. Transferable Solution Method for Parametric Optimal Control Problems

Many practical optimal control applications require repeatedly solving problems where system dynamics remain fixed but objectives vary. For example, in trajectory planning, the target destination may change from one instance to another, giving rise to a class of problems that must be solved repeatedly. While existing approaches (see Section IV) can achieve high accuracy on a fixed problem setting, changes in the objective typically require recomputation of solutions from scratch, leading to substantial computational overhead.

The central challenge hindering model adaptability is to efficiently approximate the family of control policies $\{\mathbf{u}^*(\cdot, \cdot; \boldsymbol{\eta})\}_{\boldsymbol{\eta}}$ without re-solving each problem instance. To address this, we propose a methodology that directly targets task variability. Our approach leverages the function encoder framework to learn a transferable policy representation, enabling efficient adaptation to new problem specifications with limited, or even no additional data.

Function Encoder Enabled Control Policies

Our key idea is to approximate the function space of parametric control policies using a function encoder. Specifically, the control policy is modeled as a linear combination of learned basis functions:

$$\mathbf{u}(\mathbf{x}, t; \boldsymbol{\eta}) \approx \mathbf{u}_{\boldsymbol{\theta}}(\mathbf{x}, t; \boldsymbol{\eta}) = \sum_{j=1}^{p} c_j(\boldsymbol{\eta}) \, \phi_j(\mathbf{x}, t; \boldsymbol{\theta}_j), \tag{7}$$

where $\{\phi_j(\cdot, \cdot; \boldsymbol{\theta}_j)\}_{j=1}^p$ are basis functions realized by neural networks with collective parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_j\}_{j=1}^p$, and $\mathbf{c}(\boldsymbol{\eta})$ are coefficients specific to the task $\boldsymbol{\eta}$.

We highlight that the key feature of this formulation is that the basis functions are learned once and are independent of the task parameter $\boldsymbol{\eta}$, thereby forming a reusable representation of the policy function space. Consequently, the problem of transferring to a new task reduces to determining the task-specific coefficients $\mathbf{c}(\boldsymbol{\eta})$. This can be achieved in a purely data-driven manner, namely given some dataset $\mathcal{D}$ comprised of $\{\mathbf{x}_i, t_i, \mathbf{u}(\mathbf{x}_i, t_i)\}_{i=1}^M$ corresponding to some task $\boldsymbol{\eta}$, we consider minimizing the LS formula

$$C(\mathbf{c}, \mathcal{D}) := \frac{1}{M} \sum_{i=1}^{M} \left\| \mathbf{u}(\mathbf{x}_i, t_i) - \sum_{j=1}^{p} c_j \, \phi_j(\mathbf{x}_i, t_i) \right\|_2^2. \tag{8}$$

```
Algorithm 1: FE Training for Control Space
  Input: Learning rate $\alpha$, regularization parameter $\lambda$,
         task dependent datasets $\{\mathcal{D}_{S_k}\}_{k=1}^{N}$, and loss
         function $C$ from (8);
1 Initialize bases $\{\phi_j\}_{j=1}^{p}$ with params $\boldsymbol{\theta} = \{\boldsymbol{\theta}_j\}_{j=1}^{p}$;
2 while not converged do
3     $L \leftarrow 0$;
4     for k = 1,2,...,N do
5         $\mathbf{c} = \arg\min_{\mathbf{c}} C(\mathbf{c}, \mathcal{D}_{S_k})$;
6         $L \leftarrow L + \|\mathbf{u}_{S_k} - \sum_{j=1}^{p} c_j \phi_j\|_{\mathcal{H}}^2$;
7     $L \leftarrow L + \lambda \sum_{j=1}^{p} \|\phi_j\|_{\mathcal{H}}^2$;
8     $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} L$;
  Output: Trained bases $\{\phi_1, \phi_2 \dots, \phi_p\}$
```

```
Algorithm 2: (Optional) Operator Network Training
  Input: Trained bases $\{\phi_j\}_{j=1}^{p}$ with parameters $\boldsymbol{\theta}$,
         task–dataset pairs $\{(\boldsymbol{\eta}_{S_k}, \mathcal{D}_{S_k})\}_{k=1}^{N}$, learning
         rate $\beta$, and loss function $C$ from (8);
1 Initialize operator net. $\psi : \boldsymbol{\eta} \mapsto \mathbb{R}^p$ with params $\gamma$;
2 while not converged do
3     $L \leftarrow 0$;
4     for k = 1,2,...,N do
5         $\mathbf{c} = \arg\min_{\mathbf{c}} C(\mathbf{c}, \mathcal{D}_{S_k})$;
6         $L \leftarrow L + \|\mathbf{c} - \psi(\boldsymbol{\eta}_{S_k})\|_2^2$;
7     $\gamma \leftarrow \gamma - \beta \nabla_{\gamma} L$;
  Output: Trained operator network $\psi$
```

over fixed bases $\{\phi_j\}$ to obtain $\mathbf{c}(\boldsymbol{\eta})$. In practice, task-dependent datasets $\mathcal{D}$ can be obtained through standard open-loop solvers, pretrained policy models, or other means.

Algorithmic Pipeline

Our methodology follows an offline–online scheme (see Figure 1) designed to balance computational efficiency with flexibility. The basis functions are learned only once during the offline phase, while task-specific adaptation takes place online by determining the coefficients $\mathbf{c}(\boldsymbol{\eta})$, either from limited observed state-action measurements or via a direct mapping from the task specification $\boldsymbol{\eta}$. This offline-online separation allows the intensive computation to be performed only once during the offline phase, leaving online adaptation lightweight and suitable for real-time control.

a) Offline Phase: FE training is in its core imitation learning. Given task parameterization $\{\boldsymbol{\eta}_1, \dots, \boldsymbol{\eta}_N\}$ and their associated datasets $\{\mathcal{D}_{S_1}, \dots, \mathcal{D}_{S_N}\}$, we train the basis functions via Algorithm 1. For simplicity, we assume that each dataset contains M labeled observations $\{((\mathbf{x}_i, t_i), \mathbf{u}_{S_k}(\mathbf{x}_i, t_i))\}_{i=1}^{M}$.

For low-dimensional and structured parameterization of tasks (e.g., target locations), inspired by operator-learning approaches such as [22], we optionally introduce an operator network $\psi : \boldsymbol{\eta} \mapsto \mathbf{c}(\boldsymbol{\eta})$ with parameters γ. This enables data-free coefficient inference during the online phase. The network is trained via a least-squares reconstruction loss using the fixed bases $\{\phi_j\}$ learned offline; see Algorithm 2.

b) Online Phase: At deployment, the trained basis functions $\{\phi_j\}_{j=1}^{p}$ are fixed, and online adaptation reduces to estimating the task-specific coefficients $\mathbf{c}(\boldsymbol{\eta})$ via:

1) **Zero-shot LS.** Given some trajectory data for the policy under new task $\boldsymbol{\eta}$, we can estimate $\mathbf{c}(\boldsymbol{\eta})$ by least-squares projection onto the learned basis functions (minimizing (8) over measurements).
2) **Zero-shot operator.** Set $\mathbf{c}(\boldsymbol{\eta}) = \psi(\boldsymbol{\eta})$ for data-free adaptation using the trained operator network.

We find that in practice, when limited measurements are available for a new task, the LS approach generally gives better performance. On the other hand, online inference is completely data-free using the operator method, though it does incur additional computation cost and requires more training data during the offline phase, and can be particularly hard when $\boldsymbol{\eta}$ is high-dimensional or complex. The justification of such trade-off often hinges on each specific problem.

We note that while Theorem 1 guarantees that the bias of the policy adaptation can be arbitrarily small, (8) is only a discrete formulation of (5), where the inner product in $\mathcal{H}$ is approximated by finite measurements. For a given task, denote the minimizers of (5) and (8) be $\mathbf{c}$ and $\mathbf{c}^\star$, respectively. They generally differ due to sampling size and measurement noise.

Theorem 2. *Under standard assumptions for LS regression with fixed design. Denote by $\Phi \in \mathbb{R}^{M \times p}$ the matrix whose i-th row is $\phi(\mathbf{x}_i, t_i)^\top$ and $B = \frac{1}{M}\Phi^\top \Phi$ the Gram matrix. Assume sub-Gaussian noise in measurements and B full rank, for any $\delta \in (0, 1)$, with probability at least $1 - \delta$,*

$$\|\mathbf{c} - \mathbf{c}^\star\|_2 \;\le\; \frac{\sigma}{\sqrt{\lambda_{\min}(B)}} \sqrt{\frac{p + \log(1/\delta)}{M}} \;=\; \mathcal{O}\big(M^{-1/2}\big),$$

for some constant σ.

The proof follows [41, Thm. 2.2, 2.3], the details of which we omit here. This result, together with Theorem 1, suggests asymptotic convergence of unbiased policy prediction as the number of bases and samples grows, validating our proposed approach in principle for both offline training and online inference steps. Empirical validation on function encoder is also available at [23].

IV. Related Work

We review approaches to optimal control problems through a local–global lens, then discuss learning-based methods that bridge these extremes, motivating our focus on transferable policy approximation across tasks.

Local Solution Methods (Trajectory Optimization)

The most common approach for solving optimal control problems is *direct transcription*, which converts the continuous problem into a finite-dimensional optimization problem that is then solved via gradient-based methods [8], [39]. Multiple shooting [9] extends this idea and often improves numerical stability on long-horizon problems. Although highly optimized solvers exist [46], [4], they provide accurate but inherently local solutions that must be recomputed whenever the initial state or objective changes.

Global Solution Methods

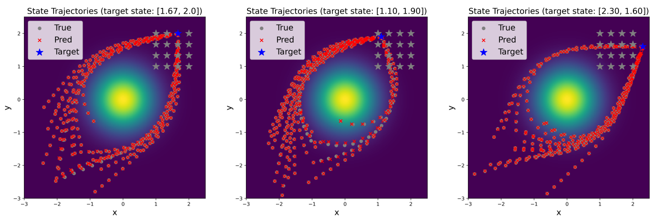
The Hamilton–Jacobi–Bellman (HJB) equation [16] characterizes the globally optimal feedback via the value function, from which the optimal control can be recovered as the minimizer of the Hamiltonian. This is appealing because it yields a single policy valid across initial conditions given the objective, rather than re-solving a new local trajectory each time. In general, solving the HJB equation requires discretizing the state–time domain [32], [37], [1], and the computational and memory costs grow exponentially with the state dimension (the "curse of dimensionality"), making mesh-based methods non-scalable in practice. Similarly, solution approaches based on multi-parametric programming [6], [2], [27] face the same issues. Special cases exist; for instance, LQR problems [3], [7] have exact solutions via the Riccati equation; it is powerful yet non-general.

Learning-based Approaches

ML amortizes solve time into an offline phase to produce feedback policies evaluable online—bridging local trajectory optimization and global HJB solvers. Notably, these learned policies are usually *semi-global*: by opting for sampling instead of meshing, they provide reliable feedback over broad regions of the state–time domain, making them scalable, though they remain task-specific in their standard form. *Data-driven approaches* such as [19], [24], [12], [13] train policies on solver-generated trajectories and can achieve fast convergence and high accuracy. *Model-based and physics-informed approaches* [18], [36], [43], [15], [35], [14], [26], [20], [10], [33] leverage OC structure (objective terms, HJB residuals, Koopman representation, etc.) to provide robustness and reduce data requirements in training, though the resulting optimization can be considerably more challenging. *Model-free RL* [44], [21] learns feedback without an explicit mathematical model but as a trade-off is often sample-inefficient and may underperform model-based approaches [45]. While ML-based approaches have seen extraordinary success in solving optimal control problems, repurposing a trained model to new tasks efficiently and accurately remains an open challenge. This work aims to address that gap.

V. Numerical Experiments

To demonstrate the breadth and generalizability of our approach, we evaluate two categories of optimal control problems differentiated by the source of task variability: changes in the terminal cost and changes in the running cost. The former is illustrated with a linear 2D trajectory planning problem (Section V-A) and a 12D quadcopter problem with nonlinear dynamics (Section V-B). The latter, motivated by



(a) seen target in training, new initial states (b) new target, new initial states (interpolation) (c) new target, new initial states (extrapolation)

Fig. 2: Generalization results for 2D trajectory planning. All cases test on new initial states, demonstrating semi-global policies that work on both seen and unseen target scenarios.

scenarios such as varying terrain and obstacles, is examined through two problems based on a nonlinear bicycle model (Section V-C). Together, these examples span low and high state dimensions as well as linear and nonlinear dynamics, providing extensive verification of our method.

Our FE implementation closely follows [23]. We use multi-head, multi-layer perceptrons (MLPs) for basis function parameterization. Exact training details vary depending on the example. For data preparation and ground truth comparison we rely on standard discretization-based open-loop solvers using SciPy or CasADi [4]. Our code is implemented in PyTorch [38] and will be released upon publication. All experiments were conducted on an NVIDIA A100 GPU.

A. 2D Path Planning with Different Targets

We first consider a 2D trajectory planning problem adapted from [29] where the goal is to find the optimal path between some initial and target states while avoiding a fixed obstacle. The initial state of the agent is not fixed but sampled from a Gaussian distribution, that is, $\mathbf{x}_0 \sim \rho = \mathcal{N}((-1.5, -1.5)^\top, 0.4 \cdot \mathbf{I}_2)$. We consider linear dynamics where $f = \mathbf{u}$ in (1), with costs

$$L(\mathbf{x}, \mathbf{u}) = \frac{1}{2}\|\mathbf{u}\|^2 + Q(\mathbf{x}), \quad G(\mathbf{x}) = 50 \cdot \|\mathbf{x} - \mathbf{y}\|^2,$$

where $\mathbf{y}$ denotes the target state and can change depending on task specification parameter $\boldsymbol{\eta}$ in (2). The function $Q(\cdot)$ models an obstacle placed between the initial and target location, given by

$$Q(\mathbf{x}) \;=\; 50\,\exp(-1.25\,\|\mathbf{x}\|^2).$$

Each task in the example corresponds to a different target state, and our objective is to recover a solution that is evaluable online for any sampled initial state and given target, including those not seen during training. We prepare the dataset with 16 different target states over a 2D grid between 1 and 2 for both x and y. For each target state, we solve the OC problem under 200 random initializations to obtain trajectories using a direct approach. Here, we set $T = 1$ and use 20 time steps for the forward integration of the dynamics.

For the FE parameterization, we use a 4-layer MLP with hidden size 256, with 100 basis functions in total. For training, we use the Adam optimizer [25] with a learning rate of 0.001, for a total of 20K steps to ensure convergence. We present the numerical results in Figure 2, including both the test results on $\mathbf{y}$ seen in the training dataset and cases beyond the training domain. We highlight that in all the tested scenarios, the learned model approximates the ground-truth control accurately without any retraining. To quantify the performance of the learned model, we also report the objective functional value as a reference in Table I. Notice that our model retains a low objective functional value with errors below 4% across all test cases. Note that in the original FE work [23] model performance is most reliable when generalizing to tasks within the convex hull of the training task distribution, here we also observe promising extrapolation results, such as in Figure 2c.

Additionally, we present quantitative comparisons between different coefficient estimation methods in Table I. The LS approach achieves higher accuracy during online evaluation. Although the operator approach requires additional data and computation for offline training, it offers the advantage of minimal computational cost during the online phase.

We highlight that our approach is fundamentally different from any local solution methods, as the feedback policy can be applied for arbitrary inputs once trained, demonstrated in Figure 3.

TABLE I: Average true and predicted objective loss across evaluations for the 2D path planning problem. The average is taken across different target states and initial conditions.

Evaluation	True Objective Loss	Predicted Objective Loss	Inference Method
seen target	4.8609	4.8737	LS
new target interpolation	4.8295	4.8608	
new target extrapolation	4.8782	5.0751	
seen target	4.8609	4.9625	operator
new target interpolation	4.8295	4.9578	
new target extrapolation	4.8782	5.3214	

B. Quadcopter Path Planning with Different Targets

We consider controlling a quadcopter under complex dynamics; see [36]. Here, the problem is modeled with a 12 dimensional state variable $\mathbf{x} \in \mathbb{R}^{12}$ and control inputs consist of a thrust u and torques $(\tau_\psi, \tau_\theta, \tau_\varphi)$. The objective is to steer the system from a stationary initial state randomly sampled

$$\mathbf{x}_0 = [\xi_1 - 2,\, \xi_2 - 2,\, \xi_3 - 2,\, 0,\, \dots,\, 0]^\top,\ \xi_i \sim \mathcal{N}(0, 0.5^2),$$

for $i = 1, 2, 3$, to a target state $\mathbf{y} = [y_1, y_2, y_3, 0, \dots, 0]^\top \in \mathbb{R}^{12}$, where y_1, y_2, y_3 can be chosen differently. The system dynamics are given as

$$\begin{cases} \ddot{x} = \frac{u}{m}\left(\sin(\psi)\sin(\varphi) + \cos(\psi)\sin(\theta)\cos(\varphi)\right) \\ \ddot{y} = \frac{u}{m}\left(-\cos(\psi)\sin(\varphi) + \sin(\psi)\sin(\theta)\cos(\varphi)\right) \\ \ddot{z} = \frac{u}{m}\cos(\theta)\cos(\varphi) - g \\ \ddot{\psi} = \tau_\psi \\ \ddot{\theta} = \tau_\theta \\ \ddot{\varphi} = \tau_\varphi \end{cases}, \quad (9)$$

Here (x, y, z) is the position, (ψ, θ, φ) is the orientation, $g = 9.8$ is the gravitational constant, and $m = 1$ is the mass. The corresponding first-order system to (9) can be easily derived [36]. We consider the objective function given as

$$L(\mathbf{x}, \mathbf{u}) = \frac{1}{2}\|\mathbf{u}\|^2, \quad G(\mathbf{x}) = 500 \cdot \|\mathbf{x} - \mathbf{y}\|^2,$$

where the penalty on the terminal cost is chosen to be large, following [36], to ensure that different, even far-away targets can always be reached.

For this example, our goal is to recover transferable solutions with respect to different target states. The high dimensionality of the state space and the strong nonlinearity of the dynamics make the problem particularly challenging. For data preparation, we use a direct transcription approach for solving trajectory-based solutions given each initial and target states. We fix $T = 2$ and use 50 time steps for forward integration. For function encoder training, we consider 64 different target states generated over a uniform grid between 1 and 4 in the x-, y-, and z-directions. We then sample initial states and solve for 80 different trajectories. Additionally, a separate test dataset is prepared with 27 new tasks; in each case, the test set includes 25 trajectories from different initial states.

We use the same model architecture and optimization setting as in Section V-A, with 100K iterations until convergence. In Figure 4 we visualize the generalization results of the trained model under a new target state. The model can accurately predict controls and guide the quadcopter to the target location from multiple initializations, demonstrating the effectiveness of our proposed approach. Quantitative results are displayed in Table II. We note that despite the high dimensionality and nonlinearity of the problem, our learned model can achieve high accuracy across different tasks. In fact, under the zero-shot LS inference approach, our policy incurs only 0.4% error in objective value over all 27 tasks tested. Similar to the findings in Section V-A, the LS approach at inference yields more accurate predictions, with additional data requirements as the trade-off.

TABLE II: Average true and predicted objective loss across different tasks for the Quadcopter path planning problem, here the average is taken with respect to both different target states and sampled initial conditions.

Evaluation	True Objective Loss	Predicted Objective Loss	Inference Method
seen target	276.9546	278.8933	LS
new target	274.3089	275.4412	
seen target	276.9546	278.4307	operator
new target	274.3089	294.5627	

C. Bicycle Control Under Different Obstacle Configurations

We now consider the scenario where the task specification determines the running cost L. This can be common in situations where traversing through different terrain types is needed. Changes to the running cost can substantially

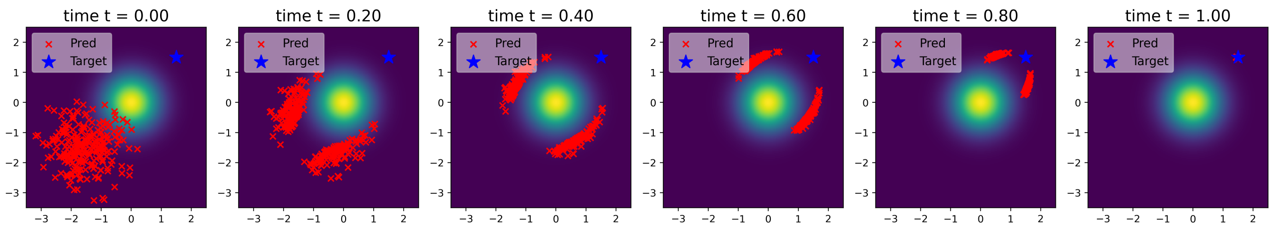
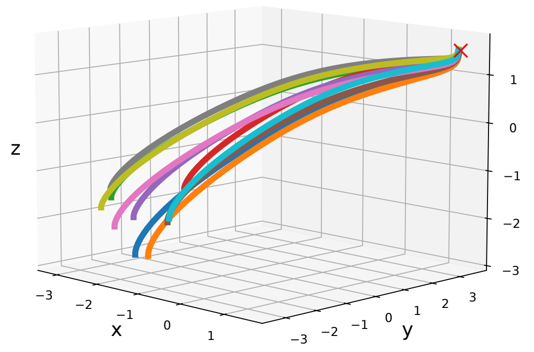
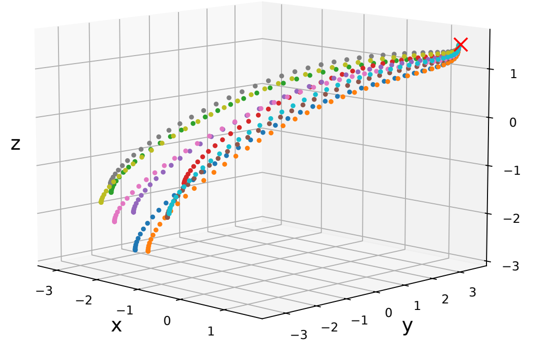


Fig. 3: Visualization of the learned control policy. We generate a sample of size 256 following the specified distribution for the initial state. The plot illustrates how they traverse over time following the learned control, demonstrating consistent performance across the state space.

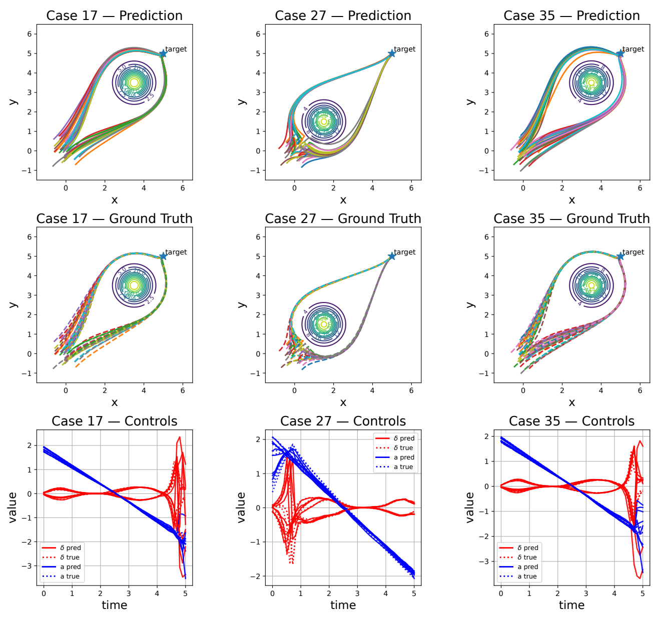


(a) Predictions



(b) Ground Truth

Fig. 4: Generalization results for the quadcopter path planning problem. Visualization for a new target $\mathbf{y} = [1.5, 3.5, 1.5, 0, \ldots, 0]^\top$ not seen during training. We compare model predictions to corresponding true solutions.



(a) obstacle config 1 (b) obstacle config 2 (c) obstacle config 3

Fig. 5: Visualization of the *three worst performing scenarios* of the learned controller tested over new problem settings for the single obstacle example. Top: Predicted solutions over different initial states. Middle: Ground truth solution calculated w.r.t. the same initial states. Bottom: Visualization of the controls, here for clarity, we only show a few instances.

alter the control behavior, making these types of problems particularly challenging. Unlike terminal objectives such as target locations, which often admit compact representations, running costs are commonly specified as high-dimensional cost maps over the full domain [28], making them difficult to compress into a form suitable for zero-shot transfer. Hence, in this section, we focus only on the zero-shot LS approach.

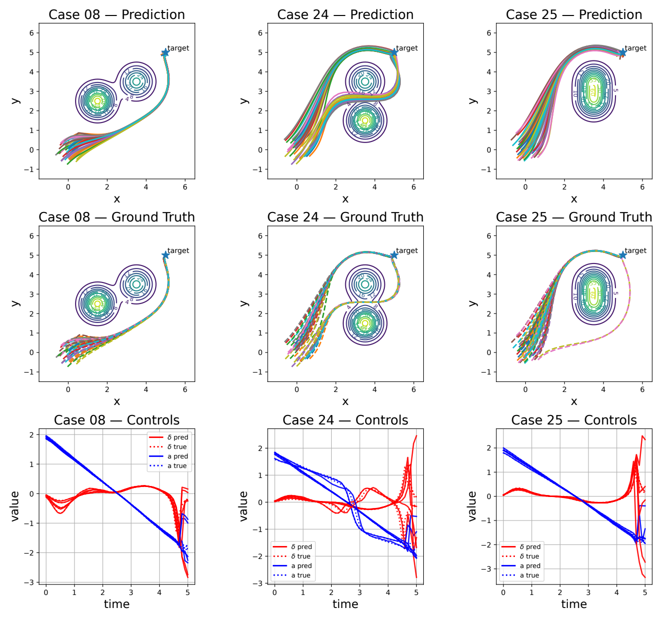
For the experiment, we study the motion of a bicycle model. The goal is to find the optimal steering and acceleration to guide the bicycle from a starting point to a target position and orientation over a fixed time horizon. The state of the system at any time t is given by a 4-dimensional vector $\mathbf{x} \in \mathbb{R}^4$, written as $\mathbf{x}(t) = \left[x(t), y(t), \theta(t), v(t)\right]^\top$, where (x, y) is the 2D position of the bicycle's rear axle, θ is the heading angle (orientation), and v is the forward velocity. The control input at t is a 2-dimensional vector $\mathbf{u}(t) \in \mathbb{R}^2$ defined as $\mathbf{u}(t) = \left[\delta(t), a(t)\right]^\top$, where $\delta(t)$ is the steering angle of the front wheel and $a(t)$ is the acceleration. The dynamics of the bicycle are described by the following.

$$f = \left[v(t)\cos\theta(t),\ v(t)\sin\theta(t),\ \frac{v(t)}{l}\tan\delta(t),\ a(t)\right]^\top, \tag{10}$$

where $l = 0.5$ is the length of the bicycle's wheelbase. For this example, while the terminal cost stays quadratic, the running cost consists of penalization on both the control and the state at t, which reads

$$L(\mathbf{x}, \mathbf{u}) = \frac{1}{2}\|\mathbf{u}\|^2 + Q(\mathbf{x}), \quad G(\mathbf{x}) = 50 \cdot \|\mathbf{x} - \mathbf{y}\|^2,$$

with Q denoting the state penalty given some obstacle. For this example, we use a time horizon $t \in [0, 5]$, the initial



(a) obstacle config 1 (b) obstacle config 2 (c) obstacle config 3

Fig. 6: Visualization of *the three worst performing scenarios* of the learned controller tested over new problem settings for the double obstacle example. Top: Predicted solutions over different initial states. Middle: Ground truth solution calculated w.r.t. the same initial states. Bottom: Visualization of the controls, here for clarity, we only show a few instances.

state $\mathbf{x}_0$ is sampled from a Gaussian distribution centered at the origin, that is,

$$\mathbf{x}_0 = [\xi_1, \xi_2, \frac{\pi}{4}, 0], \quad \xi_1, \xi_2 \sim \mathcal{N}(0, 0.35^2).$$

We fix the target state at $\mathbf{y} = [5, 5, \frac{\pi}{4}, 0]^\top$ for this example.

Solving the control problem can be difficult given the highly nonlinear dynamics of the model. For each task, we define a different obstacle configuration that can change the controller behavior, increasing the complexity of the learning problem. For demonstration purposes, we model the obstacle using a Gaussian formulation

$$Q(\mathbf{x}; A, \boldsymbol{\mu}, \sigma) = A \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \boldsymbol{\mu}\|^2\right).$$

Here the parameters A, $\boldsymbol{\mu}$ and σ control the position and shape of the obstacle and vary from each individual task. We generate data for optimal control solutions of 144 different obstacle settings by randomly selecting $A \in \{30, 40, 50\}$, $\mu_1, \mu_2 \in \{1, 2, 3, 4\}$ and $\sigma \in \{0.3, 0.5, 0.7\}$. For each scenario, we simulate 100 different trajectories for training. Another 36 new cases in obstacle configuration are reserved for testing, each also containing 100 different trajectories. We note that for the scenarios where obstacle placement is close to the initial or target states, sharp changes in the ground truth controls are to be expected. We test not only the generalizability but also the robustness of our approach through this example, as the learned model has to account for both smooth and non-smooth policies.

TABLE III: Numerical results for the obstacle generalization experiments using the bicycle model. We display the prediction and ground truth for the cumulative control and obstacle cost, as well as terminal state deviation measured in both x and y direction, all results are evaluated and averaged over multiple new problem setting unseen in training.

Evaluation	Control Cost	Obstacle Cost	Term. State Deviation
Single obstacle			
Prediction	2.6789	0.0463	0.0012
Ground Truth	2.6480	0.0385	0.00005
Double obstacles			
Prediction	3.0370	0.5750	0.0046
Ground Truth	2.9668	0.5286	0.00004

We illustrate the numerical results in Table III. Note that the learned model transfers accurately to new problem settings using only limited data for inference. In particular, the controller can learn to avoid the obstacle for arbitrary placement and reach the target state with high precision, demonstrating the capability of our proposed approach. We present qualitative results in Figure 5. Here, we display the results for 3 worst performing subproblems out of all tested cases. Notice that even in the worst performing cases, the learned model can provide reasonable and accurate guidance to the system, as demonstrated in the close match between the predicted and ground truth trajectories.

Double Obstacle Experiment: To further demonstrate the applicability of our method, we conduct additional experimentation with increasing difficulty. In this example, instead of modeling the terrain using a simple Gaussian obstacle, we introduce a non-overlapping secondary obstacle to the problem. Using the same physics in (10), we define

$$Q(\mathbf{x}; \{A_i, \boldsymbol{\mu}_i, \sigma_i\}_{i=1}^2) = \sum_{i=1}^{2} A_i \exp\left(-\frac{1}{2\sigma_i^2}\|\mathbf{x} - \boldsymbol{\mu}_i\|^2\right).$$

Consider $A_1, A_2 \in \{30, 50\}$, $\sigma_1, \sigma_2 \in \{0.3, 0.5\}$ and the obstacle placement randomly selected over a 3×3 grid between 1.5 and 3.5. We prepare the dataset consisting of 576 different obstacle configurations, we split them into a 544/32 training/test split after random shuffling. For each scenario, the dataset contains 100 trajectories. We note that the secondary obstacle largely increases the difficulty of the problem, as new pathing options lead to different controls; we increase the FE model size to account for this.

We display the corresponding numerical results in Table III, trained model under our approach remains accurate despite the increase in problem complexity. We display the model predictions and the corresponding true solution in Figure 6. Here we again only visualize the worst performing cases to show the effectiveness of our approach. We highlight that the new pathing options are correctly captured, and the model performance remains stable even when the underlying ground truth solution exhibits shock-like behavior.

VI. Conclusions

To summarize, we proposed an FE-based framework for policy transfer in parametric optimal control problems. The framework enables zero-shot adaptation to new tasks with or without data and demonstrates reliable performance across diverse examples. Future work will explore extensions to multi-agent systems with interacting dynamics.

References

[1] Y. Achdou, G. Barles, H. Ishii, G. L. Litvinov, P. Loreti, and N. Tchou. *Hamilton-Jacobi equations: approximations, numerical analysis and applications*, volume 10. Springer, 2013.

[2] A. Alessio and A. Bemporad. *A Survey on Explicit Model Predictive Control, in Nonlinear Model Predictive Control: Towards New Challenging Applications*. Springer Berlin Heidelberg, 2009.

[3] B. D. Anderson and J. B. Moore. *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

[4] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 2018.

[5] M. Bardi, I. C. Dolcetta, et al. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*, volume 12. Springer, 1997.

[6] A. Bemporad, F. Borrelli, and M. Morari. Model predictive control based on linear programming - the explicit solution. *IEEE Transactions on Automatic Control*, 47(12):1974–1985, 2002.

[7] D. Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 4. Athena scientific, 2012.

[8] J. T. Betts. *Practical methods for optimal control and estimation using nonlinear programming*. SIAM, 2010.

[9] H. G. Bock and K.-J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984.

[10] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1):411–444, 2022.

[11] P. Cannarsa and C. Sinestrari. *Semiconcave Functions, Hamilton-Jacobi Equations, and Optimal Control*. Springer, 2004.

[12] S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari. Approximating explicit model predictive control using constrained neural networks. In *2018 Annual American control conference (ACC)*, pages 1520–1527. IEEE, 2018.

[13] J. Coulson, J. Lygeros, and F. Dörfler. Data-enabled predictive control: In the shallows of the DeePC. In *2019 18th European control conference (ECC)*, pages 307–312. IEEE, 2019.

[14] P. Donti, M. Roderick, M. Fazlyab, and J. Z. Kolter. Enforcing robust control guarantees within neural network policies. In *International Conference on Learning Representations*, 2021.

[15] J. Drgoňa, A. Tuor, and D. Vrabie. Learning constrained parametric differentiable predictive control policies with guarantees. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 54(6), 2024.

[16] W. H. Fleming and H. M. Soner. *Controlled Markov processes and viscosity solutions*. Springer, 2006.

[17] G. H. Golub, P. C. Hansen, and D. P. O'Leary. Tikhonov regularization and total least squares. *SIAM Journal on Matrix Analysis and Applications*, 21(1):185–194, 1999.

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

[19] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer. Learning an approximate model predictive controller with guarantees. *IEEE Control Systems Letters*, 2(3):543–548, 2018.

[20] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1), 2020.

[21] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[22] T. Ingebrand, A. J. Thorpe, S. Goswami, K. Kumar, and U. Topcu. Basis-to-basis operator learning using function encoders. *Computer Methods in Applied Mechanics and Engineering*, 435:117646, 2025.

[23] T. Ingebrand, A. J. Thorpe, and U. Topcu. Function encoders: A principled approach to transfer learning in hilbert spaces. *arXiv preprint arXiv:2501.18373*, 2025.

[24] B. Karg and S. Lucia. Efficient representation and approximation of model predictive control laws via deep learning. *IEEE Transactions on Cybernetics*, 50(9):3866–3878, 2020.

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] M. Korda and I. Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018.

[27] M. Kvasnica and M. Fikar. Clipping-based complexity reduction in explicit MPC. *IEEE Transactions on Automatic Control*, 57(7), 2012.

[28] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

[29] X. Li, D. Verma, and L. Ruthotto. A neural network approach for stochastic optimal control. *SIAM Journal on Scientific Computing*, 46(5):C535–C556, 2024.

[30] D. Liberzon. *Calculus of variations and optimal control theory: a concise introduction*. Princeton university press, 2011.

[31] D. Liu, Q. Wei, D. Wang, X. Yang, and H. Li. *Adaptive dynamic programming with applications in optimal control*. Springer, 2017.

[32] X.-D. Liu, S. Osher, and T. Chan. Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, 115(1):200–212, 1994.

[33] J. Matschek, J. Bethge, and R. Findeisen. Safe machine-learning-supported model predictive force and motion control in robotics. *IEEE Transactions on Control Systems Technology*, 31(6):2380–2392, 2023.

[34] M. Morari, C. E. Garcia, and D. M. Prett. Model predictive control: Theory and practice. *IFAC Proceedings Volumes*, 21(4):1–12, 1988. IFAC Workshop on Model Based Process Control.

[35] T. X. Nghiem, J. Drgoňa, C. Jones, Z. Nagy, R. Schwan, B. Dey, A. Chakrabarty, S. Di Cairano, J. A. Paulson, A. Carron, M. N. Zeilinger, W. Shaw Cortez, and D. L. Vrabie. Physics-informed machine learning for modeling and control of dynamical systems. In *American Control Conference (ACC)*, pages 3735–3750, 2023.

[36] D. Onken, L. Nurbekyan, X. Li, S. W. Fung, S. Osher, and L. Ruthotto. A neural network approach for high-dimensional optimal control applied to multiagent path finding. *IEEE Transactions on Control Systems Technology*, 31(1):235–251, 2022.

[37] S. Osher and C.-W. Shu. High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations. *SIAM Journal on Numerical Analysis*, 28(4):907–922, 1991.

[38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8024–8035, 2019.

[39] A. V. Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135(1):497–528, 2009.

[40] J. B. Rawlings, D. Q. Mayne, and M. Diehl. Model predictive control: Theory and design. *Nob Hill Publishing*, 2017.

[41] P. Rigollet and J.-C. Hütter. High-dimensional statistics. *arXiv preprint arXiv:2310.19244*, 2023.

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

[43] D. Soudbakhsh, A. M. Annaswamy, Y. Wang, S. L. Brunton, J. Gaudio, H. Hussain, D. Vrabie, J. Drgona, and D. Filev. Data-driven control: Theory and applications. In *2023 American Control Conference (ACC)*, pages 1922–1939. IEEE, 2023.

[44] R. S. Sutton, A. G. Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[45] D. Verma, N. Winovich, L. Ruthotto, and B. van Bloemen Waanders. Neural network approaches for parameterized optimal control. *Foundations of Data Science*, 7(1):363–385, 2025.

[46] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl. acados – a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 2021.