Adjoint-based machine learning for active flow control

Xuemin Liu [*] and Jonathan F. MacArt [†]
Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, Indiana 46556, USA

(Received 17 July 2023; accepted 15 November 2023; published 9 January 2024)

We develop neural-network active flow controllers using a deep learning partial differential equation augmentation method (DPM). The end-to-end sensitivities for optimization are computed using adjoints of the governing equations without restriction on the terms that may appear in the objective function. In one-dimensional Burgers' examples with analytic (manufactured) control functions, DPM-based control is comparably effective to standard supervised learning for in-sample solutions and more effective for out-of-sample solutions, i.e., with different analytic control functions. The influence of the optimization time interval and neutral-network width is analyzed, the results of which influence algorithm design and hyperparameter choice, balancing control efficacy with computational cost. We subsequently develop adjoint-based controllers for two flow scenarios. First, we compare the drag-reduction performance and optimization cost of adjoint-based controllers and deep reinforcement learning (DRL) -based controllers for two-dimensional, incompressible, confined flow over a cylinder at Re = 100, with control achieved by synthetic body forces along the cylinder boundary. The required model complexity for the DRL-based controller is 4229 times that required for the DPM-based controller. In these tests, the DPM-based controller is 4.85 times more effective and 63.2 times less computationally intensive to train than the DRL-based controller. Second, we test DPM-based control for compressible, unconfined flow over a cylinder and extrapolate the controller to out-of-sample Reynolds numbers. We also train a simplified, steady, offline controller based on the DPM control law. Both online (DPM) and offline (steady) controllers stabilize the vortex shedding with a 99% drag reduction, demonstrating the robustness of the learning approach. For out-of-sample flows (Re = {50, 200, 300, 400}), both the online and offline controllers successfully reduce drag and stabilize vortex shedding, indicating that the DPM-based approach results in a stable model. A key attractive feature is the flexibility of adjoint-based optimization, which permits optimization over arbitrarily defined control laws without the need to match *a priori* known functions.

DOI: 10.1103/PhysRevFluids.9.013901

I. INTRODUCTION

Altering the natural dynamics of flows via control is desirable for engineering applications [1]. Examples include drag reduction to reduce aircraft fuel consumption [2,3] and structural damage prevention by suppressing flow-induced oscillations [4]. Flow control can be broadly classified into passive and active methods. Passive flow control (PFC) requires no external energy, relying on inherent characteristics of the flow system or the use of passive devices, while active flow control (AFC) requires energy inputs. Typical PFC techniques include geometric modifications, for example

[*] xliu24@nd.edu
[†] jmacart@nd.edu

trailing-edge flaps on airfoils to reduce nose-down pitching moments [5], or surface roughness designed to reduce skin friction drag [6]. Aerodynamic shape optimization is closely related, being typically fixed in operation, for example optimizing over computational fluid dynamics (CFD) calculations to increase the lift-to-drag ratio [7,8]. Though passive flow control techniques are cost-effective, their control efficacy is limited and is sensitive to fouling and damage, which may not be suitable for all applications. Conversely, active flow control can offer greater flexibility, though with the requirement of actuation energy, and it has been a fixture of aerodynamics since Prandtl's pioneering work on boundary-layer separation delay via oscillatory blowing and suction [9].

AFC methods can be categorized into predetermined (open-loop) and interactive (closed-loop) control. Open-loop control involves energy expenditure without necessarily measuring the flow field. Closed-loop control instead modulates actuators using sensor measurements; recent examples include blowing/suction jets to reduce drag over two-dimensional (2D) cylinders [10,11] and hot-wire sensors and loudspeaker actuators to stabilize the wake instability [12]. Choi *et al.* [13] proposed an effective framework for turbulent flows, and other critical theoretical developments can be found in the review by Brunton and Noack [14]. Combinations of PFC and AFC have also been presented, such as using a synthetic jet actuator and porous coatings to adjust drag and lift acting on a cylinder [15]. In general, active flow control provides more advanced and effective flow control than passive techniques.

Nevertheless, developing efficient AFC strategies remains a challenge [16]. Machine learning (ML) with artificial neural networks (ANNs) has received significant attention, for it is well suited to optimization and control problems involving black-box or multimodal cost functions [17]. One approach is to construct surrogate models for the flow dynamics using data-driven ML methods; examples include predicting drag and lift by training convolutional neural networks from CFD simulations [18], learning a multifidelity approximator for flow simulations [19], and reduced-order modeling of flow dynamics [20,21].

Another popular branch of ML for AFC utilizes deep reinforcement learning (DRL), which models an agent interacting with its environment so as to maximize the cost function based on a Markov decision process. DRL is widely used for complex decision-making problems (originally associated with games), and it has been applied to an increasing number of physical systems. In the past few years, DRL has been applied to laminar drag reduction using synthetic blowing and suction [10], shape optimization of airfoils [22], and turbulent drag reduction in channel flows via blowing/suction controlled by velocity measurements [23] and the wall shear stress [24]. Other algorithms have been combined with DRL to help discover better control policies; for example, sparse proximal policy optimization with covariance matrix adaptation (S-PPO-CMA) has been used to optimize sensor layouts for drag reduction [25] and to augment RL to suppress vortex shedding [26]. Pino *et al.* [27] compared the AFC performance of multiple ML methods including generic programming, deep deterministic policy gradient (a DRL variant), Lipschitz global optimization, and Bayesian optimization. More recent applications can be found in the review by Vignon *et al.* [28]. DRL is attractive for AFC applications—it requires only a properly defined flow environment, with no need for intrusive solver modification. However, many challenges remain due to DRL's sample inefficiency, difficulty of designing reward functions, and lack of stability and convergence theories [29]. To improve efficiency, many DRL applications are deliberately limited to narrow search spaces, which helps the agent focus on the most relevant states and actions [30,31]. For example, Wang *et al.* [32] projected sparse instantaneous measurements and past data into a higher-dimensional "dynamic feature" space to compensate for the low measurement dimensionality.

Gradient-based optimization can also be used to develop active flow controllers. It requires gradients of a cost function $L(\mathbf{u}, \vec{\theta})$ to be computed with respect to control parameters $\vec{\theta}$,

$$\nabla_\theta L(\mathbf{u}, \vec{\theta}) = \frac{\partial L}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \vec{\theta}} + \frac{\partial L}{\partial \vec{\theta}}, \tag{1}$$

TABLE I. Summary of numerical experiments, optimization objectives, solvers, and software packages.

Section	Experiment	Objective	Solver	Discretization	Optimization Algorithm	NN optimization toolkit
Sec. II C	Burgers' equation	MMS	In-house 1D	FDM	*A priori* ML Adjoint DPM	*PyTorch*
Sec. III	Confined cylinder	Control	*FeniCS*	FEM	DRL-PPO Adjoint DPM	*TensorForce* *Dolfin-Adjoint*
Sec. IV	Unconfined cylinder	Control	*PyFlowCL*	FDM	Adjoint DPM	*PyTorch*

where **u** are the flow variables that implicitly depend on $\vec{\theta}$. For systems governed by partial differential equations (PDEs), **u** is often high-dimensional, making $\partial\mathbf{u}/\partial\vec{\theta}$ computationally prohibitive to compute directly, especially when $\vec{\theta}$ is also high-dimensional. Adjoint-based methods, which we pursue, enable $\nabla_\theta L$ to be calculated independently of the dimension of $\vec{\theta}$ [33]. Adjoint methods can be broadly classified into continuous and discrete approaches: in the continuous approach, the optimization problem is first stated in continuous form, then discretized, while the discrete approach first discretizes the forward problem, then poses the optimization problem in discrete form.

Adjoint-based optimization originates from dynamic programming methods in optimal control theory [34]. Optimization and control are therefore closely related, and their boundaries are becoming less distinct with increasingly capable computers [35]. Applications of adjoint-based methods in aerodynamics and flow control include aircraft shape optimization to increase lift-to-drag ratio [8,36,37], topology optimization for unsteady incompressible fluid flows [38], rotating-cylinder drag reduction [39,40], multimode Rayleigh-Taylor instability suppression [41], and high-speed flow control [42]. More examples, including separation control, enhanced mixing, and noise suppression, can be found in the reviews by Collis *et al.* [16] and Kim and Bewley [43]. Most of these focus on open-loop control, in which the design parameters and/or control forces remain constant outside the optimization procedure.

A deep learning PDE augmentation method (DPM) was recently proposed to train ML models using adjoint-based, PDE-constrained optimization. While successful for large-eddy simulation (LES) subgrid-scale models [44,45], it has not yet been applied to flow control. A key feature of the DPM is its use of adjoints to provide the end-to-end sensitivities needed for ML model optimization; thus, its trained models are constrained by the system's dynamics, at least as accurately as the dynamics are modeled by the forward PDEs (unlike DRL, which approximates the system's dynamics using additional ML models). The convergence of DPM to a global minimum during optimization has been proved for linear elliptic PDEs with neural network terms [46], and further study of its application to linear and nonlinear parabolic and hyperbolic PDEs is ongoing. This work proposes a DPM approach for closed-loop active flow control: it optimizes the parameters of a neural-network flow controller by solving the flow PDEs subject to a user-defined objective function.

This paper is organized as follows. Section II introduces the DPM and DRL optimization methods, verifies the convergence of DPM-based flow controllers for a one-dimensional (1D) viscous Burgers' equation, and compares the efficacy of DPM training to that of *a priori* (offline) ML. Section III evaluates and compares the control performance and training cost of DPM- and DRL-based active flow controllers for 2D incompressible flow over a confined cylinder. Section IV analyzes the performance of DPM-trained controllers for 2D compressible flow over an unconfined cylinder, assesses the ability of the learned controller to extrapolate to higher Reynolds numbers, and tests the performance of a simplified controller inferred from the DPM controller. A summary and discussion are given in Sec. V. Table I lists the numerical experiments, optimization objectives, numerical solvers, and NN optimization toolkits used in this work.

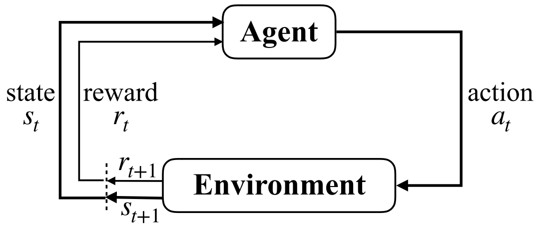

FIG. 1. DRL: interaction of the agent and environment in a Markov decision process.

II. OPTIMIZATION-BASED FLOW CONTROL

Sections II A and II B introduce the DRL and DPM algorithms and their applications to flow control. Section II C verifies and analyzes the convergence of the DPM-based control strategy for a 1D viscous Burgers' equation example.

A. Deep reinforcement learning

DRL constructs a model for an agent interacting with an environment so as to maximize an objective [47], as shown in Fig. 1. The agent is a decision-maker that maps observed states and rewards (feedback) to actions: $(s_t, r_t) \mapsto a_t$. The environment is a dynamical system that advances to the next state s_{t+1} with new feedback r_{t+1}. This is mathematically based on a Markov decision process (MDP), in which the transitions between states depend only on the current state and action (not on any previous states or actions). In general, its goal is to maximize the cumulative reward $R(\mathcal{T}) = \sum_{t=0}^{T} \gamma^t r(s_t, a_t)$ along a trajectory $\mathcal{T} = (s_0, a_0, r_0; s_1, a_1, r_1; \ldots; s_T, a_T, r_T)$ with a discount rate $\gamma \in [0, 1]$ that smooths the impact of temporally distant rewards [47]. DRL algorithms are classified into policy-based and value-based methods. In policy-based methods, the agent approximates a policy $\pi(a|s)$ that explicitly maps states to actions, where $\pi(a|s) = \Pr(a|s)$ is the probability of action a given the observed state s. In contrast, value-based methods obtain the action at time t implicitly by $a_t = \operatorname{argmax}_a V^\pi(s_{t+1})$, where $V^\pi(s) = \mathbb{E}_{\mathcal{T}\sim\pi}[R(\mathcal{T})|s]$ is a state value function, and $\mathbb{E}_{\mathcal{T}\sim\pi}[\cdots]$ refers to the ensemble mean over several stochastic trajectories $\mathcal{T}$, along which actions are randomly sampled from the policy. Actor-critic algorithms are a combination of these, where policy and value functions are learned simultaneously [48].

For comparisons to adjoint-based learning, we employ an actor-critic proximal policy optimization (PPO) algorithm to obtain an optimal policy $\pi^*(a|s)$; this approach is known to have higher data efficiency, robustness, and simplicity than other DRL methods [49]. The environment is a flow simulation (PDE solution), and the discovery of the PPO agent is done using the open-source *Tensorforce* library [50]. Pseudocode for the PPO algorithm is given in Algorithm 1 to maximize an objective function $\mathcal{L}$,

$$\mathcal{L}(\theta) = \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t),\ \ \operatorname{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon\right) A^{\pi_{\theta_k}}(s_t, a_t)\right), \tag{2}$$

where π_θ is a stochastic policy between steps k and $k+1$, A^π is the estimated advantage value corresponding to the policy, and ε is a (small) hyperparameter that determines the maximum permitted distance of the new policy from the old policy. During testing, the action is ensured to be deterministic by directly using the mean as the proposed action (i.e., disregarding any randomness). More details of this algorithm can be found in Ref. [49] (Sec. 3) and Ref. [10] (Appendix C).

While DRL is a popular method for decision-making problems, several of its limitations presented in Sec. I (including sample inefficiency, insufficient convergence of the dynamical model, and lack of stability and convergence proofs) complicate applications to flows when the gradient-descent

Algorithm 1. Actor-critic style PPO

```
Initialize actor network θ_0 and critic network φ_0
for k ← 1 to K do
    for n ← 1 to N do
        Run policy π_{θ_k} in environment for T timesteps
        Compute cumulative reward R_t = Σ_{i=t}^{T} γ^{n−i} r_i
        Estimate advantage A_t^{π_{θ_k}}(s_t, a_t) = R_t − V_{φ_k}(s_t) by evaluating the value function V_{φ_k}(s_t)
    end
    Optimize the objective function L with respect to θ, then θ_{k+1} ← θ
    Optimize value function by minimizing the regression loss (1/NT) Σ_{n=0}^{N} Σ_{t=0}^{T} (V_φ(s_t) − R_t)^2, then φ_{k+1} ← φ
end
```

search space is not sufficiently narrow. We return to the issue of sample efficiency in Sec. III, when we compare DRL to DPM for flow control.

B. Deep learning-based PDE augmentation

Rather than attempting to approximate the system dynamics with a neural network, DPM optimizes directly over the governing PDEs. The discrete solution of a system of PDEs with embedded neural-network terms, $u(t,\theta) \in \mathbb{R}^{n_d}$, is an implicit function of the neural-network parameters $\theta \in \mathbb{R}^{n_\theta}$, where $n_d = d \times n_{\text{eq}}$ for d discrete mesh nodes and n_{eq} dependent variables. For typical systems encountered in engineering and applied science, $n_d = O(10^5)$ to $O(10^9)$, and n_θ can be $O(10^5)$ or larger for modern ML models. Optimizing θ using standard gradient-descent methods would require calculating $\nabla_\theta u \in \mathbb{R}^{n_d \times n_\theta}$, which would be prohibitively expensive for typical physical systems. DPM instead calculates the gradients needed for optimization by solving adjoint variables $\hat{u}(t) \in \mathbb{R}^{n_d}$ with comparable cost to the solution of the forward PDE system.

We wish to minimize a time-averaged objective function of the PDE solution

$$\bar{J} = \int_0^\tau J(u,t,\theta)dt, \tag{3}$$

where $J(u,t,\theta)$ is an instantaneous scalar objective function, and τ is the time interval over which optimization is performed. Given initial conditions $g(u(0)) = 0$ satisfying the PDE and an instantaneous PDE residual $R(u,\dot{u},t,\theta)$, where $\dot{u} = \partial u/\partial t$, the optimization problem is

$$\min \bar{J} \text{ over}(u,\theta) \text{ subject to} R(u,\dot{u},t,\theta) = 0 \text{ and} g(u(0)) = 0. \tag{4}$$

The ANN parameters are updated from time level n to time level $n+1$ using a gradient descent step,

$$\theta_{n+1} = \theta_n - \alpha_n \nabla_{\theta_n} \bar{J}, \tag{5}$$

where α_n is the learning rate at step n. DPM computes $\nabla_\theta \bar{J}$ using the adjoint variables $\hat{u}(t)$,

$$\nabla_\theta \bar{J} = \int_0^\tau \nabla_\theta J dt = \int_0^\tau \left(\hat{u}^\top \frac{\partial R}{\partial \theta} + \frac{\partial J}{\partial \theta} \right) dt, \tag{6}$$

where $\hat{u}(t)$ satisfy the linear ordinary differential equation (ODE),

$$\frac{d\hat{u}^\top}{dt} = \hat{u}^\top \frac{\partial R}{\partial u} + \frac{\partial J}{\partial u}^\top, \tag{7}$$

which is solved backward in time from $t = \tau$ to $t = 0$ with initial condition $\hat{u}(\tau) = 0$. In (7), the Jacobian $\partial R/\partial u$ can be challenging to obtain in general; we compute it using automatic differentiation

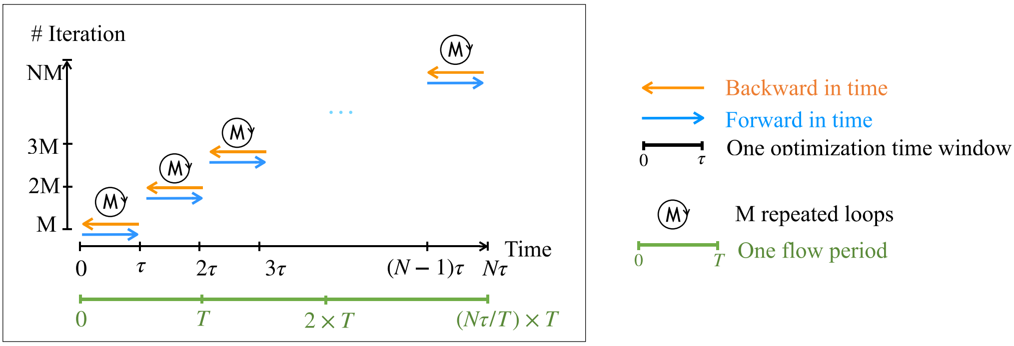

FIG. 2. Illustration of the DPM optimization steps in Algorithm 2, with $\tau = T/2$ as an example.

(AD) over the forward solution using the *PyTorch* library [51]. AD works by sequentially applying the chain rule of differentiation to evaluate the gradient of a function with respect to its inputs.

Numerical stability often requires a system's characteristic time T to be discretized over many time steps. Solving (7) requires checkpointed solutions $u(t, \theta)$ over $t \in [0, \tau]$ at intermediate time steps, with available computer memory often dictating the maximum ratio τ/T. This is an important consideration for flows, for the maximum τ/T can be significantly less than unity. Figure 2 illustrates the decomposition of multiple characteristic times T into N optimization windows of size $\tau = T/2$ (for example), each of which is optimized M times for a total of NM optimization iterations. The pseudocode for the DPM algorithm is given in Algorithm 2.

C. Example: Burgers' equation control

We now demonstrate DPM-based control of the viscous Burgers' equation with the control objective provided by an analytic solution. The Burgers' equation has a similar nonlinearity to the Navier–Stokes equation, hence it is widely used as a numerical test [52]. We prescribe an analytic solution and obtain analytic control terms using the method of manufactured solutions (MMS), obtain approximate control terms using DPM and *a priori* ML, and compare the learned controllers to the analytic controller.

Algorithm 2. DPM

Initialize ANN with initial parameters θ_0. Set the time interval for optimization τ and total number of optimization steps NM

for $n \leftarrow 0$ **to** N **do**
  for $m \leftarrow 0$ **to** M **do**
    for $i \leftarrow n\tau$ **to** $(n+1)\tau$ **do**
      Compute the flow variables u_i by simulating the PDE $R(u, \dot{u}, \theta, t) = 0$ forward in time
    end
    Initialize adjoint variables $\hat{u}_{(n+1)\tau} = 0$
    for $j \leftarrow (n+1)\tau$ **to** $n\tau$ **do**
      Compute the adjoint variables $\hat{u}_j$ backward in time by solving Eq. (7)
    end
    Compute gradients $\nabla_\theta \bar{J}$ via Eq. (6)
    Optimize the controller $\theta \leftarrow \theta'$ via Eq. (5)
  end
end

1. *Control framework*

Consider the one-dimensional viscous Burgers' equation with a source term,

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} + S(\mathbf{u}, \theta), \tag{8}$$

with domain $x \in [0, L]$, $L = 1$, solution $u(x, t)$, initial condition $u_0(x)$, and periodic boundary conditions. The source term has inputs $\mathbf{u}$ and, in the case of modeled source terms, tunable parameters θ. A constant viscosity $\nu = 0.002$ is used. MMS prescribes a target solution $u^e(x, t)$ for which an analytic source term $S^e(x, t)$ may be obtained. Modeling the source term using a neural network enables direct comparison of ML-based control methods to the exact control term.

We aim to learn a class of controllers $S(\mathbf{u}, \theta)$ that modify the dynamics of (8) to match those of a first-order linear advection equation,

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0, \tag{9}$$

with matching domain, initial condition, and boundary conditions as (8) and constant advective velocity a, giving a characteristic timescale $T = L/a$. The analytic source term for a prescribed u^e is therefore

$$S^e = (u^e - a)\frac{\partial u^e}{\partial x} - \nu\frac{\partial^2 u^e}{\partial x^2}. \tag{10}$$

We represent the modeled source term $S(\mathbf{u}, \theta)$ using a simple, four-layer, fully connected neural network with 200 hidden units per layer. The neural network has the same architecture as one used previously for LES subgrid modeling [44]. At a mesh node x_i, the neural network inputs are the spatially local solution $u_i = u(x_i)$ and its nearest neighbors $\mathbf{u} = (u_{i-3}, u_{i-2}, u_{i-1}, u_i, u_{i+1}, u_{i+2}, u_{i+3})$, and its output is the source term at the mesh node x_i.

We consider two approaches to optimize the controller parameters θ. Both minimize a time-integrated loss $\bar{J} = \int_0^\tau J dt$ but differ in the choice of instantaneous loss J and hence how the optimization is performed.

(i) *A priori ML* minimizes $J(\theta) = \frac{1}{2}[S(\mathbf{u}^e, \theta) - S^e]^\top[S(\mathbf{u}^e, \theta) - S^e]$, where the modeled $S(\mathbf{u}^e, \theta)$ is evaluated using the *a priori* known exact solution $\mathbf{u}^e$. This can be performed offline, without solving (8), hence it is computationally inexpensive.

(ii) *DPM (adjoint-based ML)* minimizes $J(u(\theta)) = \frac{1}{2}[u(\theta) - u^e]^\top[u(\theta) - u^e]$ by optimizing over (8), where we explicitly denote the dependence of the computed solutions u on θ. This is done using the adjoint method described in Sec. II B with residual $R = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} - S(\mathbf{u}, \theta)$.

For model training, we consider a Gaussian initial profile $u_0(x) = \exp[-(x - x_0)^2/(2\sigma^2)]$, where $\sigma = 0.1$ and $x_0 = L/2$. For numerical tests, the computational domain is discretized using a uniform mesh of 256 grid points, and all derivatives are calculated using second-order central differences. Time is advanced using the forward Euler method with step size $\Delta t = 2 \times 10^{-4}$, which corresponds to a CFL number of approximately 0.05. The learning rate is initialized as $\alpha_0 = 10^{-3}$ and decays as $\alpha_n = 0.997^n\alpha_0$ for *a priori* and $\alpha_n = 0.997^{\lfloor n/5 \rfloor}\alpha_0$ for DPM.

2. **A priori** *ML versus DPM control performance*

We first compare the in- and out-of-sample performance of *a priori* ML and DPM control, where the DPM-trained controllers use a fixed training window $\tau = 100\Delta t$. The influence of the training window τ is assessed in Sec. II C 3 for in-sample cases.

The training data set for *a priori* ML models comprises source-term snapshots $S^e(x, i\Delta t)$, $i = 1, \ldots, 5000$, which covers $t/T \in [0, 1]$ for $\Delta t = 2 \times 10^{-4}$. This process requires 71 min on one NVIDIA RTX-6000 GPU for 2000 training iterations. Optimization of the DPM models targets $u^e(x, i\Delta t)$, $i = 1, \ldots, 5000$ using the same learning rate and exponential scheduler. For the same number of optimization iterations, DPM training requires 964 min (13× the computational cost of

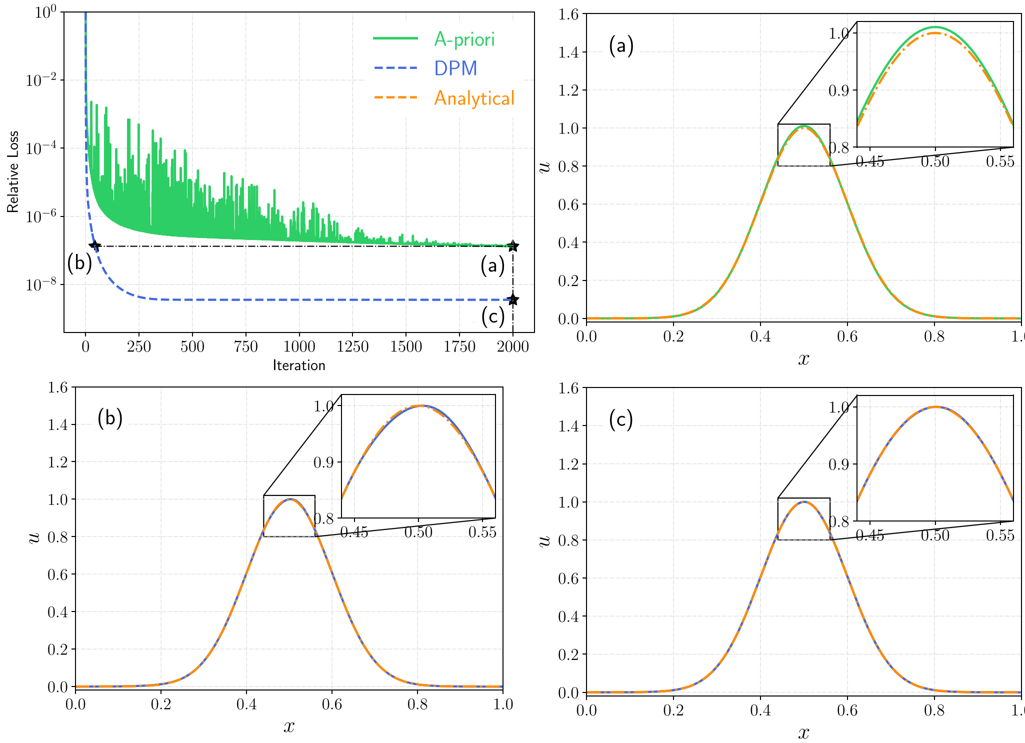

FIG. 3. Top left: Relative training loss for *a priori* and DPM-trained models $\epsilon_{\text{rel}} = J_n/J_0$, where J_0 is the norm of targets. Remaining quadrants: at $t/T = 1$, (a) Solution using best-case *a priori* ML model trained for 2000 iterations; (b) Solution using DPM model with matching training relative loss (43 iterations); (c) Solution using DPM model trained for 2000 iterations.

a priori training). This increase is understandable, for DPM training includes the cost of solving the governing equations.

Figure 3 compares the relative training loss of *a priori* and DPM-trained models, $\epsilon_{\text{rel}} = J_n/J_0$, where J_n is the per-iteration loss, and J_0 is the norm of targets $J_0(\theta) = \frac{1}{2}(S^e)^\top(S^e)$ for DRL and $J_0(u(\theta)) = \frac{1}{2}(u^e)^\top(u^e)$ for DPM. Since *a priori* ML and DPM use different loss functions (Sec. II C 1), the relative loss is the appropriate metric to compare the relative convergence of trained models. The *a priori*–trained model converges to $\epsilon_{\text{rel}} = 1.34 \times 10^{-7}$ over 2000 training iterations. The DPM-trained model achieves $\epsilon_{\text{rel}} = O(10^{-9})$ convergence in as few as 250 iterations and converges to $\epsilon_{\text{rel}} = 2.71 \times 10^{-9}$ over 2000 iterations. To obtain $\epsilon_{\text{rel}} = O(10^{-7})$, matching the *a priori* model's convergence after 2000 iterations, the DPM model requires only 43 training iterations. The wall-time for this was 20.7 min—approximately one-third the training time of the *a priori* model.

Three trained models, indicated in the top-left quadrant of Fig. 3, are chosen for testing: (a) an *a priori* model trained for 2000 iterations, (b) a DPM model trained for 43 iterations, matching the minimum *a priori* relative training error, and (c) a DPM model trained for 2000 iterations. Instantaneous snapshots of the controlled solutions for each of these three models at $t/T = 1$ are shown in the remaining quadrants of Fig. 3. At $t/T = 1$, models (a) and (b) have $O(10^{-5})$ *a posteriori* testing error, while the error for model (c) is $O(10^{-7})$ and is visually indistinguishable from the analytical target solution.

Models (a) and (c) are now tested over a longer window $t_{\text{test}}/T = 10$. Figure 4 displays the space-time evolution of the uncontrolled (baseline) and controlled viscous Burgers' equation for

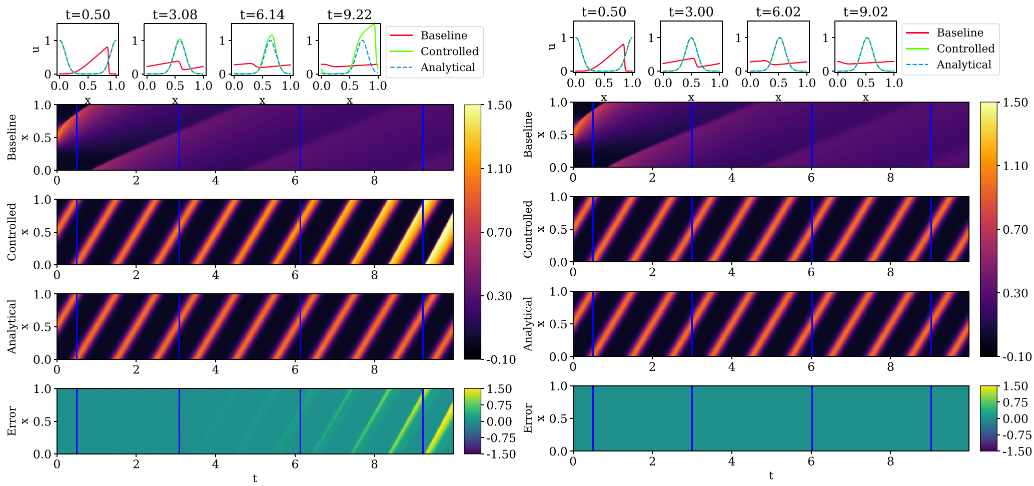

FIG. 4. In-sample performance: *a priori* ML (left) and DPM (right) controllers targeting an advecting Gaussian function. Top to bottom in each group: Instantaneous snapshots, uncontrolled baseline solution, controlled Burgers solution, analytical target, and error in the controlled solution.

these models over this longer test duration. The DPM-controlled solution is comparable to the target solution, even well beyond the training time window, while the quality of the *a priori* control degrades for $t/T > 2$. The space-time mean-squared error (MSE) of the controlled solution,

$$\epsilon = \frac{1}{Lt_{\text{test}}} \iint (u - u^e)\, dt\, dx, \tag{11}$$

is 4.37 % for the *a priori* ML controller and 0.000 02% for the DPM controller for this in-sample test.

Out-of-sample testing targets are generated using Fourier series with random coefficients (a_l, b_l, c) [53],

$$\begin{aligned} u_0(x) &= \frac{2w(x)}{\max_x |w(x)|} + c, \\ w(x) &= a_0 + \sum_{l=1}^{L} a_l \sin(2\pi l x) + b_l \cos(2\pi l x), \\ a_l, b_l &\sim \mathcal{N}(0, 1),\ L = 4,\ \text{and } c \sim \mathcal{U}(0, 1), \end{aligned} \tag{12}$$

where $\mathcal{N}(0, 1)$ is the normal distribution with mean 0 and variance 1, and $\mathcal{U}(0, 1)$ is a uniform distribution with minimum 0 and maximum 1. This is a stringent test, for the functional form of the corresponding exact source term is significantly different from that of the in-sample Gaussian target. Figure 5 shows control results for the out-of-sample target. The controlled systems gradually diverge for both control methods, occurring more slowly for DPM than for *a priori* ML, with mean-squared errors 58.03% for *a priori* ML control and 8.70% for DPM control. The latter has almost $7\times$ better stability for this out-of-sample test.

3. Influence of the DPM training window

The choice of the DPM training window τ can strongly affect training convergence and long-time stability. Despite this, previous applications of DPM [44,45] were limited to fixed τ. We now assess its influence by training DPM models over $\tau/T \in [2 \times 10^{-4}, 1]$ for advecting Gaussian targets.

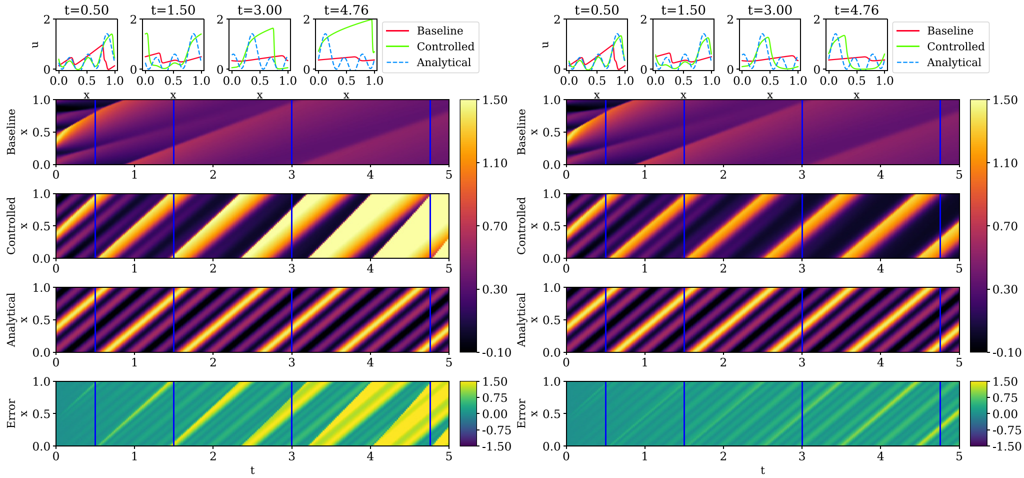

FIG. 5. Out-of-sample performance: *a priori* ML (left) and DPM (right) controllers targeting a multiscale Fourier function. Top to bottom in each group: Instantaneous snapshots, uncontrolled baseline solution, controlled Burgers solution, analytical target, and error in the controlled solution.

Figure 6 displays the in-sample MSE (11) of these solutions versus τ/T; ten randomly initialized training processes were used to obtain confidence intervals $\pm\sigma$. Instantaneous snapshots of controlled solutions and forcing terms for $\tau/T = 0.02$ are also shown. The DPM control error is minimized for τ/T ranging from 10^{-3} ($\tau/\Delta t = 5$) to 10^{-1} ($\tau/\Delta t = 500$). Large testing errors occur for $\tau/T < 10^{-3}$ ($\tau/\Delta t < 5$), for which the observation windows are too short to fully observe the system's dynamics, and for $\tau/T \approx 1$ ($\tau/\Delta t > 5000$), for which the adjoint magnitudes are significantly larger than the Jacobian determinant, leading to roundoff error accumulation [54].

Figure 6 also compares the in-sample control performance of *a priori*–trained neural networks targeting (10) with numerically evaluated derivatives (labeled "A-priori MMS"), *a priori*–trained models targeting the analytic source term, and the numerically solved MMS solution without a neural network [i.e., evaluating (10) numerically during simulation]. Their errors are generally lower than the DPM error, though it is important to recall the DPM controller's better out-of-sample performance (Fig. 5). Furthermore, controllers with analytic source terms are extremely rare in practice, rendering *a priori* control difficult or impossible, while DPM is capable of targeting any quantity derived from the flow solution.

For the subsequent applications to laminar flow control, the DPM training window is set to $\tau/\Delta t = 10$, $\tau/T = 0.016$ for the incompressible Navier-Stokes equations (Sec. III) and $\tau/\Delta t = 50$, $\tau/T = 0.005$ for the compressible equations (Sec. IV). These were found to provide adequate training for this viscous Burgers' equation example while maintaining reasonable computational cost.

III. DPM VERSUS DRL FOR ACTIVE FLOW CONTROL

We compare the control performance and training cost of DPM- and DRL-based flow controllers for drag reduction in two-dimensional (2D), incompressible, laminar flows over a confined cylinder on unstructured meshes. The flow configuration and numerical solver are identical to those used for DRL control by Rabault *et al.* [10,55]; the unstructured-mesh DPM implementation is new.

A. Numerical simulation of confined cylinder flow

The flow configuration is adapted from the benchmark computations of Schäfer *et al.* [56], in which a cylinder of diameter D is situated in a two-dimensional domain of size $(L, H) =$

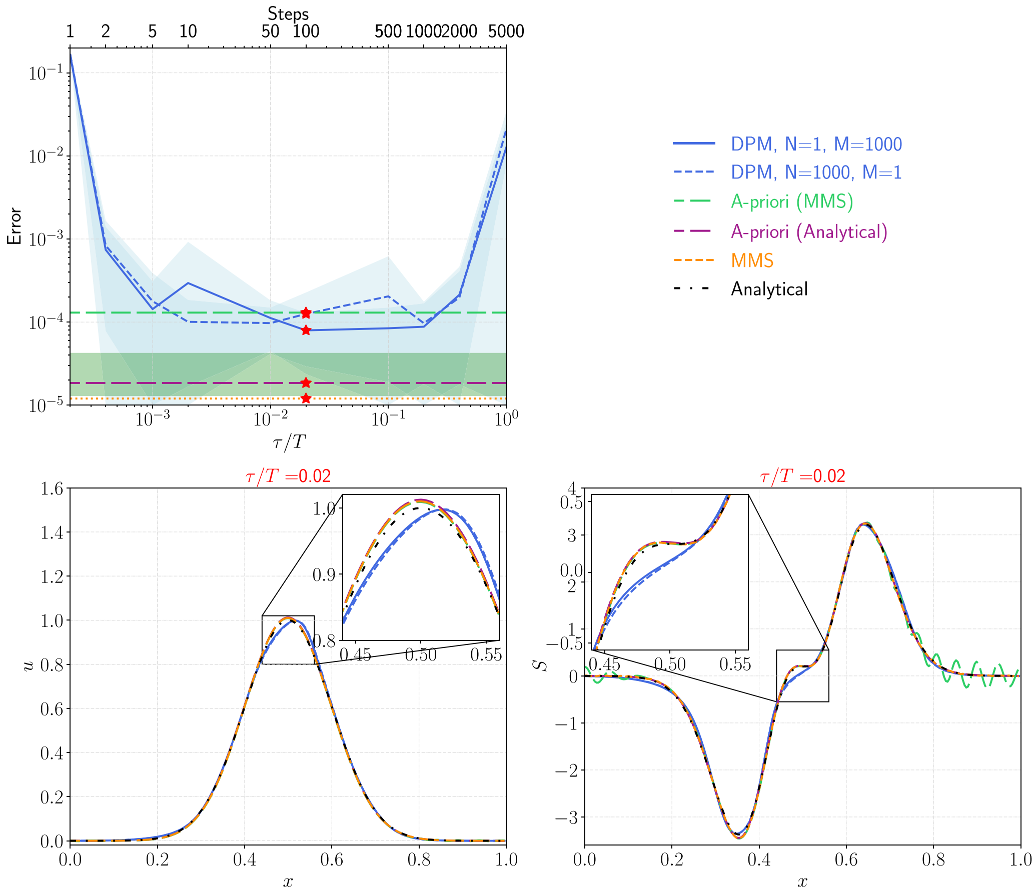

FIG. 6. Top: Error of DPM, *a priori*, and MMS controlled solutions, showing means and confidence intervals $\pm\sigma$ obtained using 10 random processes. Bottom: Instantaneous comparisons of analytical and controlled solutions (left) and control terms (right) at $t/T = 1$; DPM models used training window $\tau/T = 0.02$.

$(22D, 4.1D)$ that is open at its streamwise ends and bounded by no-slip walls in the cross-stream direction. The coordinate system origin $(x, y) = (0, 0)$ is placed at the cylinder center, and the cylinder is shifted vertically $0.05D$ from the channel centerline. This geometry and its boundaries are depicted in Fig. 7.

The flow is governed by the unsteady, incompressible, dimensionless Navier-Stokes equations,

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} + \frac{\partial p}{\partial x_i} - \frac{1}{\mathrm{Re}} \frac{\partial^2 u_i}{\partial x_j \partial x_j} = f_i(p, \theta), \tag{13}$$

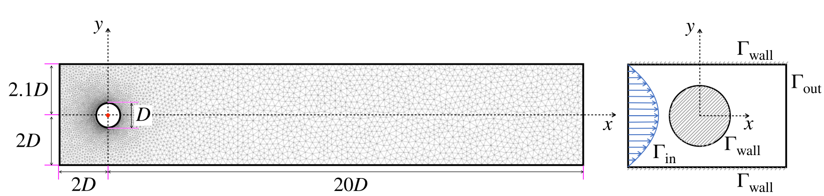

FIG. 7. Left: Geometrical configuration of 2D confined flow, illustrating the unstructured mesh density. Right: Boundary conditions (not to scale).

$$\frac{\partial u_k}{\partial x_k} = 0, \tag{14}$$

where x_j and t are the dimensionless space and time coordinates, $u_i(x_j, t)$ and $p(x_j, t)$ are the dimensionless flow velocity and pressure, and $f_i(p; \theta)$ are control functions of the local pressure with model parameters θ (described in Sec. III B). The dimensional length, velocity, and timescales are D, $\bar{U}$, and $D/\bar{U}$, respectively, where $\bar{U}$ is the bulk velocity. All models are trained for Reynolds number $\mathrm{Re} = \bar{U}D/\nu = 100$ flow, where $\nu = \mu/\rho$ is the kinematic viscosity, μ is the constant dynamic viscosity, and ρ is the constant density. In Sec. IV, the assumptions of constant viscosity and density are relaxed for compressible flows, and extrapolation of the learned models to higher Reynolds numbers is tested.

No-slip boundary conditions ($u = v = 0$) are imposed at the top and bottom walls and along the cylinder surface. The inflow velocity profile at Γ_{in} is

$$u_{\Gamma_{\mathrm{in}}} = u_1(x = -2D, y) = -4U_m(y - 2.1D)(y + 2D)/H^2, \tag{15}$$

where

$$U_m = \frac{3}{2}\bar{U} = \frac{3}{2}\frac{1}{H}\int_{-2D}^{2.1D} u_{\Gamma_{\mathrm{in}}} dy \tag{16}$$

is the centerline inlet streamwise velocity (the maximum inflow velocity). The convective outflow boundary condition on Γ_{out} assumes zero streamwise velocity derivatives at the outlet: $\partial u_1/\partial x_1|_{\Gamma_{\mathrm{out}}} = 0$.

The computational domain is discretized using an unstructured mesh of 9262 triangular cells with local refinement near the cylinder surface as shown in Fig. 7. The minimum and maximum cell diameters (twice the circumradius) are 0.001 93 and 0.0366, respectively. The flow equations are solved on this mesh using the finite-element method (FEM) and backward-Euler time integration using the *FEniCS* framework [57]. The dimensionless time step size is $\Delta t = 5 \times 10^{-4}$. More details on the flow geometry and numerical methods may be found in [10].

For analysis, the instantaneous drag force F_d and coefficient C_d at the cylinder boundary Γ_{cyl} are

$$F_d = \int_{\Gamma_{\mathrm{cyl}}} (\tau_{1j} - p\delta_{1j})n_j \, ds \quad \text{and} \tag{17a}$$

$$C_d = \frac{2F_d}{\rho\bar{U}^2 D}, \tag{17b}$$

where τ_{ij} is the shear-stress tensor, δ_{ij} is the Kronecker delta function, and n_j is the local unit normal vector at the cylinder surface. Time-averaged quantities $\langle\cdot\rangle$ are obtained by integrating over nodal values.

B. Control framework

The control objective is to minimize the time-averaged drag coefficient $\langle C_d\rangle$ by applying body forces f_i at the cylinder surface; these are designed to emulate synthetic jets as employed in recent DRL examples [10]. The body forces are centered at azimuthal angles $\theta_1 = 90^\circ$ and $\theta_2 = 270^\circ$ with width $\omega = 10^\circ$ and thickness $h = 0.1D$, shown in red in Fig. 8. On the discrete FEM grid, the body forces are applied within elements inside this control region.

The forcing terms $f_i(p; \theta)$ are approximated by feed-forward neural networks comprising either one layer with 100 hidden units (L1H100) or two layers with 512 hidden units each (L2H512). Both neural networks use hyperbolic-tangent activation functions. The NN input is the local pressure p within the control region; this is in contrast to the global velocity sensor used in recent DRL-based control [10]. The NN outputs (control actuations) are deterministic for DPM and stochastic for DRL. The two optimization methods and their respective objective functions are now discussed.

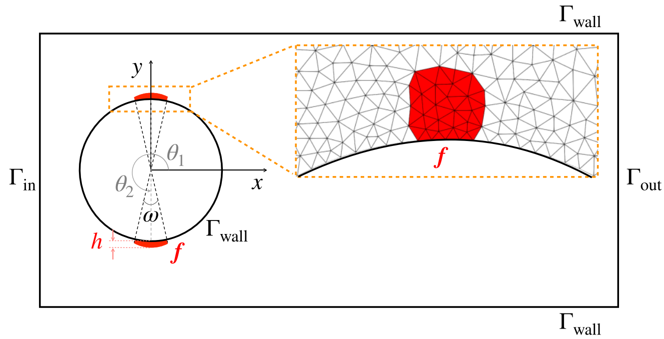

FIG. 8. Location of the body-force actuators (red) in the 2D confined cylinder configuration. The pressure sensors for actuation are located within the same regions.

1. DPM-based control framework

In the PDE-constrained DL framework, the time-averaged objective $\bar{J}$ is

$$\bar{J}(u, p, \theta) = \int_{\tau_{\rm DPM}} J(u, p, \theta)\, dt = \int_{\tau_{\rm DPM}} \|C_d(u)\|_2^2 + \beta\|f(p, \theta)\|_2^2\, dt, \tag{18}$$

where $\tau_{\rm DPM} = 10\Delta t$ is the DPM training time window, and $\beta = 0.1$ is a penalty coefficient to minimize large control energy expenditures. The forcing term f is explicitly a function of the NN parameters θ, and the velocity and pressure are implicitly functions of θ via the PDEs. The control NN is defined in the FEM space with globally uniform weights and biases (defined on a zero-degree polynomial function space), whose values are initialized by sampling from a standard normal distribution. We consider only the L1H100 network for DPM; at each node, its input is the scalar pressure, and its output is the vector body force, resulting in 402 global NN parameters (weights and biases) to be optimized. Averaging over each control cell, this amounts to 4.6 parameters per cell that require training. More details on constructing the NN in FEM space can be found in Mitusch *et al.* [58].

The adjoint variables needed for optimization are computed using *Dolfin-adjoint* [59], an open-source, discrete-adjoint solver for differentiable forward-PDE models. The NN parameters are updated every ten forward steps using L-BFGS-B [60], a limited-memory algorithm for nonlinear optimization problems subject to simple bounds. The combination of automatic differentiation for the adjoints and L-BFGS-B for parameter updates enables efficient optimization without the need to manually implement adjoint PDEs.

One DPM training iteration spans $\tau_{\rm DPM} = 10\Delta t$, and iterations are not repeated ($M = 1$). For the Re = 100 flow considered here, the testing loss of DPM flow controllers converged after approximately $N = 20$ training iterations. The convergence of the DPM-controlled mean drag coefficient, for models trained for increasing training iterations N, is shown in Fig. 9.

2. DRL-based control framework

We implement DRL control following [10] with (a) mass-source actuators replaced by momentum actuators and (b) local pressure probes at two limited regions on the cylinder boundary instead of quasiglobal velocity probes around the cylinder. As introduced in Sec. II A, the DRL control framework comprises an environment with which the controller interacts—the *FEniCS* simulation—and an agent trained using the PPO algorithm. The PPO agent is defined using *Tensorforce* [50], an open-source DRL platform based on *TensorFlow* [61]. The observed state is the pressure $s = (p)$, and the action is the body force $a = (f_1, f_2)$ within the controlled region (red cells in Fig. 8). The instantaneous reward function $r_{\hat{\tau}} = -\langle C_d\rangle_{\hat{\tau}} - 0.2|\langle C_l\rangle_{\hat{\tau}}|$ combines penalties for the time-averaged lift and drag coefficient magnitudes over a sliding time window $\hat{\tau} = 50\Delta t$.

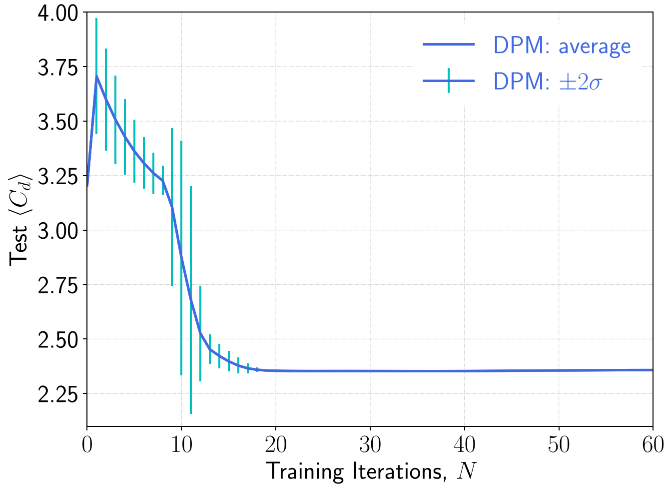

FIG. 9. Mean drag coefficients, obtained by averaging over $t_{\text{test}} = 9000\Delta t$, of DPM flow controllers optimized over different numbers of training iterations N. Test models had <0.01% change for $N > 19$. Confidence intervals are obtained from ten independently initialized tests per model.

DRL models require "actor" and "critic" networks. For the actor network, the input pressure is defined on zero-degree polynomials (a constant weight) in each controlled cell, and the output body forces are defined on two-dimensional, second-degree polynomials within each controlled cell, which are uniquely specified by 12 function weights per cell. Over the 87 controlled cells, this results in 114 244 total parameters for L1H100-based actor networks and 843 284 total parameters for L2H512-based actor networks. A given model's critic network uses the same structure as its actor network; therefore, the total number of parameters is ∼228 k for L1H100-based models and ∼1.7 m for L2H512-based models. If each controlled cell is regarded as a control point, then L1H100-based models have approximately 2.6k parameters per control point, and L2H512-based models have approximately 19.4k parameters per control point, both of which are significantly larger than the number of parameters per control point for the DPM model.

One DRL training iteration spans $\tau_{\text{DRL}} = 4000\Delta t$, corresponding to approximately 6.5 vortex shedding periods, after which the actor and critic NNs are updated over 25 optimization subiterations. A smoothly changing action function [55] is implemented to avoid flow property jumps. DRL model convergence requires approximately $N = 200$ training iterations, after which drag reduction ceases.

C. DPM versus DRL control performance

The influence of model complexity on the efficacy of the learned DPM and DRL controllers is assessed in Sec. III C 1. Having globally uniform parameters, the DPM models are $O(10^3)$–$O(10^4)$ times simpler than the locally defined DRL models. The influence of the training extent on convergence and training cost is analyzed in Sec. III C 2.

1. Control efficacy versus model complexity

This section compares an L1H100-based DPM model to L1H100- and L2H512-based DRL models trained for $N = 200$ iterations. Figure 10 shows instantaneous snapshots of velocity magnitude for the baseline Re = 100 flow, DPM-controlled flow, and L2H512 DRL-controlled flow along with the corresponding instantaneous body-force magnitudes. The DPM controller eliminates the dominant vortex shedding mode, while the DRL-controlled flow is visually similar to the baseline flow. The learned body-force terms differ significantly, with the DPM body forces having larger

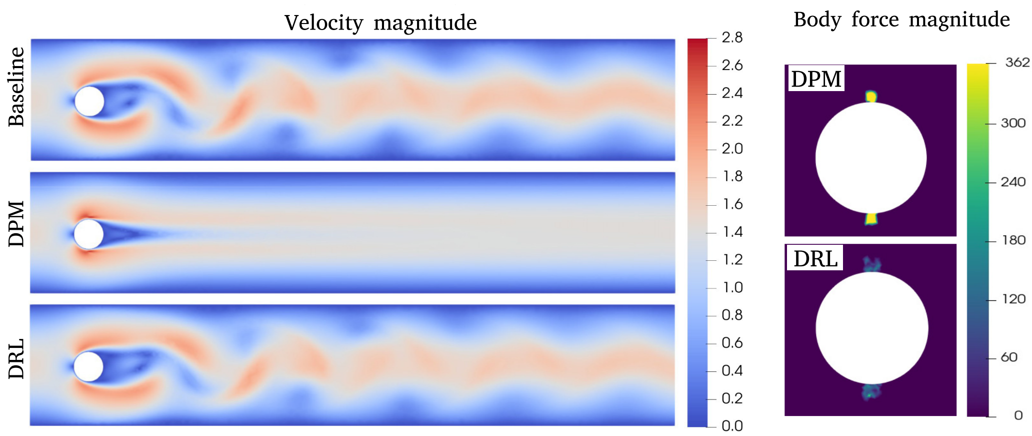

FIG. 10. Left: Instantaneous velocity-magnitude snapshots for the baseline uncontrolled Re = 100 flow, DPM-controlled flow (L1H100; 4.6 parameters per point), and DRL-controlled flow (L2H512; 19.4k parameters per point). Right: Instantaneous body-force control magnitudes.

magnitude and greater uniformity than the DRL body forces. The form of the DPM control terms for compressible flows is analyzed in Sec. IV E.

Figure 11 presents the time-series drag coefficients, time-averaged drag coefficient, and root-mean-square $(\cdot)'$ drag and lift coefficients for the uncontrolled flow, the DPM-controlled flow, and DRL-controlled flow for both networks. The efficacy of the single-layer DPM model is evident: it achieves 26.7% drag reduction and 99% RMS reduction from the baseline flow, while the two-layer DRL model achieves 5.5% drag reduction and ∼40% RMS reduction, and the single-layer DRL model is ineffective. Table II summarizes the models' control performance, including the Strouhal number reduction, where $\mathrm{St} = fD/\bar{U}$, and f is the vortex-shedding frequency obtained from the Fourier transform of the instantaneous lift coefficient. The DRL models' drag reduction performance is comparable to those of Rabault *et al.* [10,55], with the present use of momentum sources rather than mass sources being the only significant difference.

That the 402-parameter DPM model achieves approximately five times the drag reduction of the 1.7m-parameter DRL model is striking, though not entirely unanticipated. The DPM training

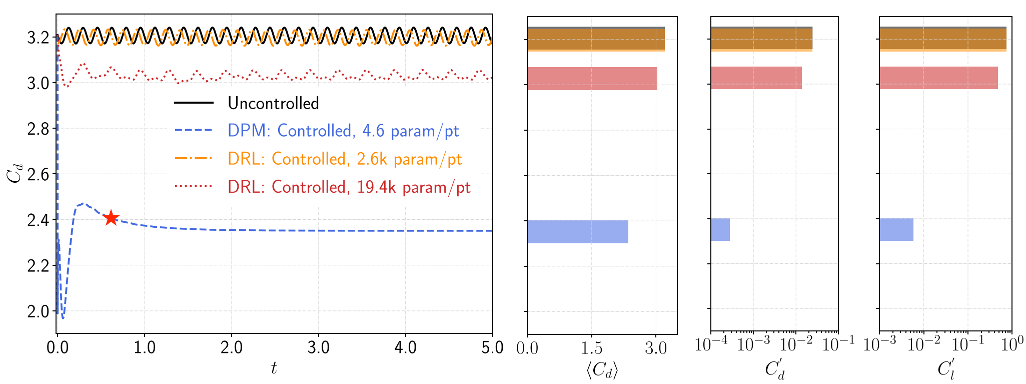

FIG. 11. Left: Time evolution of uncontrolled and (in-sample) controlled instantaneous drag coefficient. Right: Mean and RMS drag coefficient and mean lift coefficient for flow times $t \geqslant 2.5$. The red star indicates the time after which the DPM $C_d' < 5\,\%$.

TABLE II. Control performance of DPM and DRL controllers for confined cylinder flow: Time-averaged and RMS drag coefficient, RMS lift coefficient, and Strouhal number. "Mag." indicates magnitude; "Red. (%)" indicates percent reduction from the baseline.

	$\langle C_d \rangle$		C_d'		C_l'		St	
	Mag.	Red. (%)	Mag.	Red. (%)	Mag.	Red. (%)	Mag.	Red. (%)
Baseline	3.21	–	0.0245	–	0.744	–	0.32	–
DPM (L1H100)	2.35	26.7	0.0003	98.9	0.006	99.2	0.00	100.0
DRL (L1H100)	3.20	0.3	0.0239	3.1	0.735	1.1	0.32	0.0
DRL (L2H512)	3.03	5.5	0.0142	44.7	0.476	35.9	0.28	12.5

directly couples model optimization to the governing PDEs, which are in some sense the most efficient representation of the system's dynamics, though the adjoint equation's local linearization requires careful choice of the optimization window (Sec. II C 3). This leads to high "sample efficiency" of the DPM training algorithm. In contrast, DRL approximates the flow dynamics using the critic network, which is potentially expensive to train and may not always provide a flawless representation of the controlled system. The increased training cost of DRL, discussed next, and its potentially reduced out-of-sample control efficacy are prominent concerns in configurations for which adjoint-based optimization could be performed. Nevertheless, it is important to recognize that ongoing improvements to DRL algorithms (e.g., [32]) promise to increase its sample efficiency and potentially reduce its training cost.

2. Control efficacy versus training cost

The influence of training cost on DPM and DRL controller efficacy is now assessed. The significant algorithmic differences between the two methods complicate cost comparisons; in the interest of fairness, we present results for (i) converged training loss $\langle C_d \rangle_{\text{train}}$, obtained by averaging over the respective τ_{DPM} or τ_{DRL} for the final training iteration, and (ii) equal training iterations (N).

Training cost is measured as wall time on a single AMD Ryzen Threadripper 5945WX CPU core averaged over ten independent simulations.[1] DPM training required 37 s per iteration (3.7 s per Δt), and DRL training required 218 s per iteration (0.0545 s per Δt). Though the per-iteration DPM training cost is higher, largely due to the cost of constructing and solving adjoint equations at each time step, its sample efficiency enables it to converge in fewer N, or, for equivalent N, to achieve greater control efficacy.

Training convergence (Comparison 1) was deemed to occur for <5 % variance in $\langle C_d \rangle_{\text{train}}$. This occurred after $N = 55$ iterations for DPM models and $N = 200$ iterations for DRL models. The first set of rows in Table III compare the wall-time cost and mean training drag for these two models. The DPM model requires approximately 47 min for $N = 55$ iterations, resulting in $\langle C_d \rangle_{\text{train}} = 2.74$, while the DRL model requires almost 1.5 days for $N = 200$ iterations, resulting in 10 % higher testing loss of $\langle C_d \rangle_{\text{train}} = 3.01$. The testing drag, evaluated over $9000\Delta t$, is $\langle C_d \rangle_{\text{test}} = 2.95$ for DRL and $\langle C_d \rangle_{\text{test}} = 2.35$ for DPM. For DRL, $\langle C_d \rangle_{\text{test}} \approx \langle C_d \rangle_{\text{train}}$ due to its training time horizon (τ_{DRL}) spanning several vortex shedding cycles. Conversely, the DPM models achieve *lower* controlled drag coefficients in testing due to the relatively short τ_{DPM}. This is at least indicative of the typical stability of adjoint-trained deep learning models.

To reach $\langle C_d \rangle_{\text{train}} \approx 3.0$, the minimum achieved by DRL for $N = 200$, DPM requires only $N = 14$. The second set of rows in Table III (Comparison 2) shows the training costs for $N = 14$: DPM

[1]Both models' training can be parallelized and accelerated using GPUs, but their differing parallel efficiency further complicates cost comparisons, thus we compare only single-core performance.

TABLE III. Training cost and drag reduction performance for both DPM and DRL schemes, where the time and mean drag coefficients were obtained by averaging ten independent simulations. Average wall times are reported as d:hh:mm:ss.

Training	No. Parameters	Iterations N	Wall Time	$\langle C_d \rangle_{\text{train}}$	Converged?
		Comparison 1: Converged Training Loss			
DPM	402	55	33:48	2.74	Yes
DRL	1.7m	200	1:11:52:50	3.01	Yes
		Comparison 2: Equal Training Iterations			
DPM	402	14	8:38	2.98	No
DRL	1.7m	14	46:35	3.22	No

takes approximately 9 min; DRL takes 46 min and gives 8% higher $\langle C_d \rangle_{\text{train}} = 3.22$. The DRL testing drag coefficient ($\langle C_d \rangle_{\text{test}} = 3.22$) is again similar to its training drag coefficient, while the DPM testing drag coefficient ($\langle C_d \rangle_{\text{test}} = 2.42$) is again lower than its training drag coefficient.

It is difficult to understate the efficiency of adjoint-based optimization compared to deep reinforcement learning. A fully converged, adjoint-trained DPM model requires as few as $N = 20$ training iterations of duration $\tau_{\text{DPM}} = 10\Delta t$: 200 total simulation time steps. A converged DRL model—that achieves only one-fifth the drag reduction of the DPM model—requires at least $N = 200$ iterations of duration $\tau_{\text{DRL}} = 4000\Delta t$. Adjoint-based optimization therefore converges approximately *4000 times faster*, in terms of simulation time steps, than deep reinforcement learning and achieves superior control performance.

IV. ANALYSIS OF DPM CONTROL

In this section, we apply DPM for drag reduction to the control of unconfined flows over cylinders governed by the compressible Navier-Stokes equations. The numerical solver and flow configuration are introduced in Secs. IV A–IV C. DPM controllers are analyzed and tested in Secs. IV D–IV F.

A. Governing equations and numerical methods

We solve the 2D, compressible, dimensionless Navier-Stokes equations in conservative form,

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_j}{\partial x_j} = 0, \tag{19}$$

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_i u_j}{\partial x_j} + \frac{1}{\text{Ma}^2}\frac{\partial p}{\partial x_i} - \frac{1}{\text{Re}}\frac{\partial \tau_{ij}}{\partial x_j} = f_{\rho u_i}, \tag{20}$$

$$\frac{\partial \rho E}{\partial t} + \frac{\partial \rho u_j E}{\partial x_j} + \frac{1}{\text{Ma}^2}\frac{\partial p u_j}{\partial x_j} - \frac{1}{\text{Re}}\frac{\partial u_i \tau_{ij}}{\partial x_j} + \frac{1}{\text{Ma}^2\text{RePr}}\frac{\partial q_j}{\partial x_j} = u_i f_{\rho u_i}, \tag{21}$$

where ρ is the mass density, $E = e + u_i u_i/2$ is the total energy, $e = T/(\gamma \text{Ma}^2)$ is the internal energy, T is the temperature, $\gamma = 1.4$ for calorically perfect air, and dimensionless variables are obtained by normalizing dimensional quantities $\tilde{(\cdot)}$ by a reference length L, freestream velocity u_∞, density ρ_∞, pressure p_∞, and temperature T_∞:

$$t = \frac{\tilde{t} u_\infty}{L}, \quad x_i = \frac{\tilde{x}_i}{L}, \quad \rho = \frac{\tilde{\rho}}{\rho_\infty}, \quad u_i = \frac{\tilde{u}_i}{u_\infty}, \quad p = \frac{\tilde{p}}{\gamma p_\infty}, \quad T = \frac{\tilde{T}}{(\gamma - 1)T_\infty}, \quad e = \frac{\tilde{e}}{u_\infty^2}.$$

This yields scaling Mach, Reynolds, and Prandtl numbers

$$\text{Ma} = \frac{u_\infty}{\sqrt{\gamma p_\infty / \rho_\infty}}, \quad \text{Re} = \frac{\rho_\infty u_\infty L}{\mu_\infty}, \quad \text{and} \quad \text{Pr} = \frac{c_p \mu_\infty}{\lambda_\infty},$$

where μ_∞ is a dimensional reference viscosity, and c_p is the specific heat at constant pressure. The viscous-stress tensor and the heat-flux vector are

$$\tau_{ij} = \mu\left[\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right] + \mu_B \frac{\partial u_k}{\partial x_k}\delta_{ij}, \tag{22}$$

$$q_j = -\mu \frac{\partial T}{\partial x_j}, \tag{23}$$

with $\mu = 1.0$. The bulk viscosity μ_B is neglected for the $\mathrm{Ma} = 0.1$ flows considered here. The system is closed by the dimensionless ideal gas law, $p = (\gamma - 1)\rho T/\gamma$. The momentum actuators $f_{\rho u_i}$ are defined in Sec. IV C.

The governing equations are solved on generalized curvilinear coordinates using the transform $(x, y) \mapsto (\xi, \eta)$. Derivatives in the computational plane (ξ, η) are calculated using standard fourth-order central-difference schemes and lower-order, one-sided schemes at domain boundaries. Second derivatives are obtained by repeated application of first derivatives. Time is advanced using the fourth-order Runge-Kutta method.

Sixth-order implicit spatial filters [62] are applied at every time step to remove spurious oscillations arising from the use of central differences. A class of $(2N + 1)$-point, maximally tridiagonal filtering schemes may be obtained from

$$\zeta_f \bar{\phi}_{i-1} + \bar{\phi}_i + \zeta_f \bar{\phi}_{i+1} = \sum_{n=0}^{N} \frac{a_n}{2}(\phi_{i+n} + \phi_{i-n}), \tag{24}$$

where $\bar{\phi}$ is the filtered quantity. The filter stencil reduces to lower-order, one-sided stencils near domain boundaries. Values of a_n for filters of different orders may be found in [62]; we use $\zeta_f = 0.495$.

B. Simulation of unconfined cylinder flow

The physical-space grid for an unconfined cylindrical domain is defined in terms of the computational-plane coordinates,

$$x(\xi, \eta) = R(\eta)\cos(\xi), \tag{25a}$$

$$y(\xi, \eta) = R(\eta)\sin(\xi), \tag{25b}$$

with a uniform computational mesh

$$\xi_i = i \times d\xi, \quad d\xi = 2\pi/N_\xi, \quad i = 0, \dots, N_\xi - 1,$$

$$\eta_j = j \times d\eta, \quad d\eta = 1/(N_\eta - 1), \quad j = 0, \dots, N_\eta.$$

The radial grid is nonuniform using a hyperbolic-tangent stretching,

$$R(\eta) = h\frac{\tanh\left(s_R(\eta - 1)\right)}{\tanh(s_R)} + R_{\min} + h, \tag{26}$$

where $h = R_{\max} - R_{\min}$, $R_{\min}$ is the cylinder radius, $R_{\max}$ is the grid boundary radius, and s_R is the stretching parameter. The grid is shown in Fig. 12 for $N_\xi = 512$, $N_\eta = 512$, $R_{\min} = 0.5$, $R_{\max} = 150$, and $s_R = 6.5$. The minimum mesh spacings in the radial and azimuthal directions are $d\eta = 0.0029$ and $R_{\min} d\xi = 0.0061$. The azimuthal direction has a periodic boundary (the narrow gap in Fig. 12).

No-slip boundary conditions are imposed on the cylinder wall. Far-field boundaries are imposed using absorbing layers; these add source terms $\sigma(\mathbf{Q}_{\mathrm{ref}} - \mathbf{Q})$ to the governing equations, with conserved quantities $\mathbf{Q} = (\rho, \rho u_i, \rho E)$ and a boundary proximity function

$$\sigma(x, y) = \begin{cases} \alpha\left(1 - \frac{d((x,y),\partial^e\Omega)}{\delta_\alpha}\right)^p & \text{for } d((x, y), \partial^e\Omega) < \delta_\alpha, \\ 0 & \text{otherwise.} \end{cases} \tag{27}$$

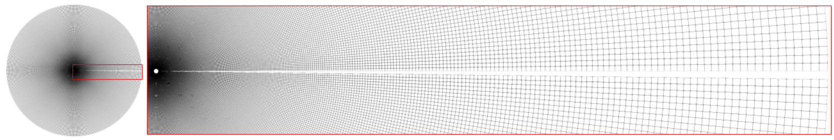

FIG. 12. Left: Mesh nodes for the full domain. Right: Zoomed view of the downstream domain sector.

In (27), $d((x, y), \delta^e\Omega)$ is the distance from a point (x, y) to the boundary $\delta^e\Omega$, $\delta_\alpha = 1$ is the absorbing layer thickness, $\alpha = 10$ is the source-term strength, and $p = 3$ is its polynomial order. Dirichlet conditions $\mathbf{Q} = \mathbf{Q}_{\text{ref}}$ are then applied on $\partial^e\Omega$.

The dimensionless time-step size is $\Delta t = 3 \times 10^{-4}$, which corresponds to CFL numbers between 0.35 and 0.45. The flow is first simulated for $t \in [0, 300]$, after which time-averaged statistics (e.g., mean drag coefficient $\langle C_d \rangle$ and Strouhal number St) are computed for $t \in [300, 360]$, representing approximately 19 vortex-shedding cycles. Figure 13 displays the overall satisfactory agreement of the present computations with published data [63–67] for $\langle C_d \rangle$ and St for Reynolds numbers Re = [50, 100, 200, 300, 400].

C. DPM controller

The objective function used here is (18) with actuator-penalty coefficient $\beta = 10^{-5}$. Different from Sec. III, the body forces are now applied along the downwind cylinder boundary rather than the upper and lower cylinder surfaces.

The source terms $\mathbf{f}(u, p; \theta) = [f_{\rho u}, f_{\rho v}]$ are active within the region $\xi \in [-45^\circ, 45^\circ]$, $\eta = 0$. We model $\mathbf{f}$ using a neural network with parameters θ and inputs comprising the local velocity and pressure $(u, p)(\xi \in [0^\circ, 45^\circ], \eta = 0)$. The neural-network outputs are $\mathbf{f}(\xi \in [0^\circ, 45^\circ], \eta = 0)$, which are mirrored symmetrically ($f_{\rho u}$) and antisymmetrically ($f_{\rho v}$) about $\xi = 0^\circ$. The fully connected neural network has one input layer, two consecutive hidden layers, a gate layer that performs an elementwise multiplication on the first hidden layer, and one linear output layer. The hidden layers have 200 units each and use rectified linear unit (ReLU) activation functions. This control framework and neural-network structure are illustrated in Fig. 14.

The controller is trained for Re = 100 flow, starting from a statistically stationary initial condition $\mathbf{Q}_0 = (\rho_0, \rho u_0, \rho v_0, \rho E_0)$. The neural-network parameters are updated over optimization windows $\tau = 50\Delta t$; one training epoch spans $N = 1000$ optimization windows without repeats

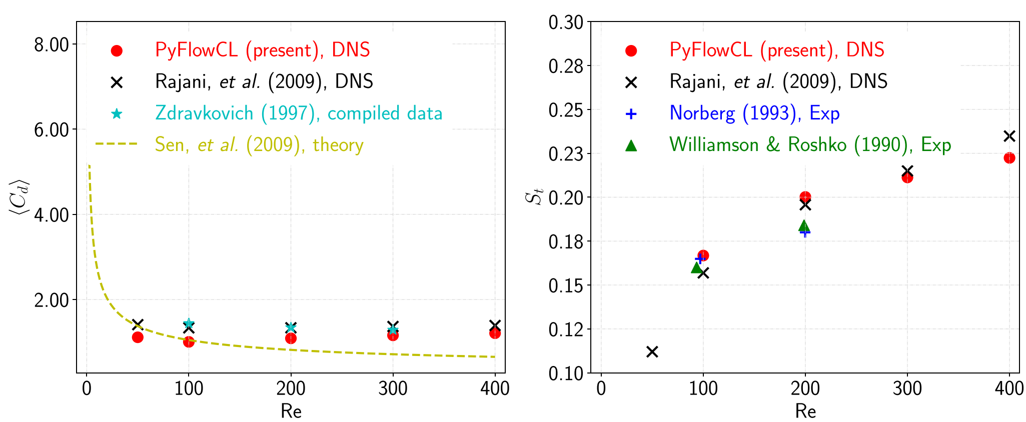

FIG. 13. Mean drag coefficient $\langle C_d \rangle$ (left) and Strouhal number S_t (right) at different Reynolds numbers. Data in Ref. [63] are compiled from experimental, DNS, and analytical theories.

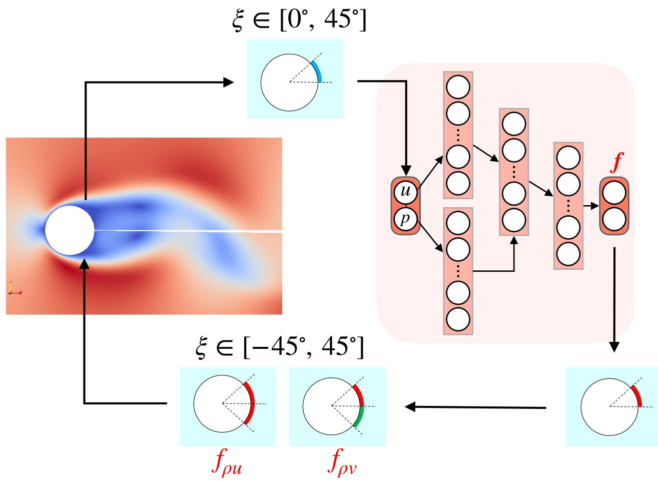

FIG. 14. Illustration of the DPM-based active-control framework.

($M = 1$). The influence of the number of training epochs is assessed in Sec. IV D. Parameter gradients are updated using the *RMSprop* optimizer with constant learning rate $\alpha = 10^{-4}$. Adjoints needed for optimization are computed using automatic differentiation over the discretized forward equations using the *PyTorch* package [51]. One training epoch required approximately 8.4 wall-time hours on one 64-core AMD EPYC 7532 server node.

D. Control Effectiveness

The DPM controller is trained for two epochs and tested for $t_{\text{test}} = 60$ time units. Figure 15 illustrates the qualitative changes to the velocity magnitude, pressure, and vorticity in the actively controlled flow. New vortical structures originate in the controlled region, representing counter-rotating vorticity to the dominant vortex-shedding modes, and the vortical shedding in the wake is visually suppressed. The actuation forces cause higher pressures within the controlled region; these partially offset the upstream stagnation pressure and reduce the overall pressure drag (the main drag source for this flow).

The time-averaged drag $\langle C_d \rangle$ and root-mean-square drag C_d', lift coefficient C_l', and separation angle θ_{sep}' are presented in Fig. 16 and Table IV for DPM-trained models for one, one and a half, and two training epochs. Also shown is the performance of simplified, constant actuator $\mathbf{f}_{\text{simp}} = \langle \mathbf{f}(u, p; \theta) \rangle_{0.05 t_{\text{test}}}$ obtained by averaging the "online" actuator $\mathbf{f}(u, p; \theta)$ over $0.05 t_{\text{test}}$. Sig-

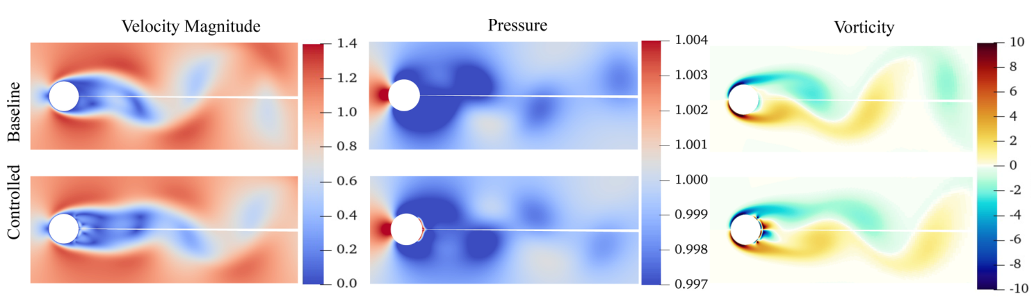

FIG. 15. Velocity magnitude, pressure, and vorticity magnitude snapshots for the baseline and online-DPM controlled flow. The controlled fields for the simplified-DPM controller are visually similar to those for the online-DPM control.

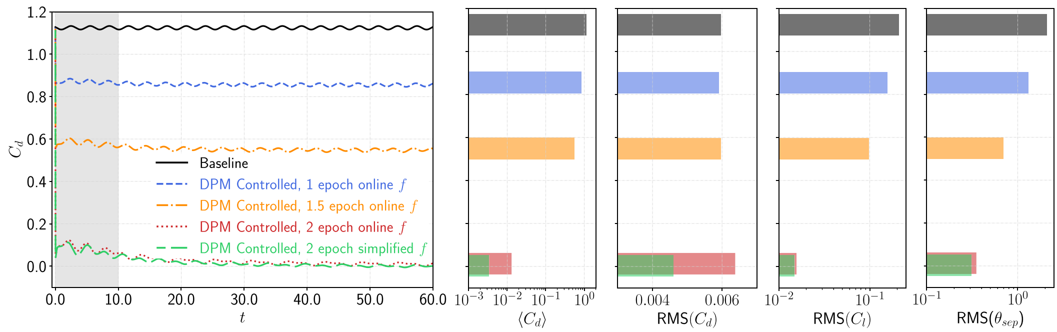

FIG. 16. Time-series drag coefficient C_d and statistical measures of flows controlled by online and simplified DPM actuators.

nificantly, the simplified actuator does not have any inputs. The variance in the online actuator is low (Sec. IV E), hence this constant actuator is surprisingly effective.

The DPM-trained controllers significantly reduce the instantaneous drag, with a 95% reduction occurring within $t = 0.5$ for the two-epoch-trained model and reaching 99% reduction around $t = 20$. The long-time control performance improves between one and two training epochs but does not change significantly with further training. The models trained here achieve overall superior drag-reduction to the DPM model in Sec. III due to the wider control region and deeper neural network. There are also evident effects of the controllers on flow oscillations. The lift fluctuations C_l' are reduced with increasing training epochs, with approximately $O(10)$ reduction after two training epochs. Likewise, the RMS fluctuations in the flow-separation angle θ_{sep}' decrease with training. While these properties were not directly targeted for optimization, they could be included in loss functions if desired, for example for resonance and flutter control in fluid-structure interaction problems.

E. Interpretability

Figure 17 shows snapshots of the velocity and pressure at the cylinder surface in the uncontrolled and controlled flow (using the two-epoch-trained model) at instants spaced by $3\Delta t$ after the onset of control actuation. The baseline flow has clear oscillations of the maximum velocity $u_{\max}$ and minimum pressure $p_{\min}$ that alternate between peaks at $\xi = 100^\circ$ and 260° (slightly upstream of the vertical points). The two-epoch-trained model suppresses the flow oscillations within $t = 6\Delta t$, leading to a quasistable flow pattern in which $u_{\max} \approx 0.61$ and $p_{\min} \approx 0.0995$. Within the control region ($\xi \in [-45^\circ, 45^\circ]$), the velocity is negative and the pressure is positive, which together cause

TABLE IV. Control performance of online and simplified (constant) DPM controllers for unconfined cylinder flow: Time-averaged and RMS drag coefficient, RMS lift coefficient, and separation angle RMS. "Mag." indicates magnitude; "Red. (%)" indicates percent reduction from the baseline.

		$\langle C_d \rangle$		C_d'		C_l'		θ_{sep}'	
	# Ep.	Mag.	Red. (%)	Mag.	Red. (%)	Mag.	Red. (%)	Mag.	Red. (%)
Uncontrolled	–	1.12	–	0.00596	–	0.2096	–	2.09	–
	1	0.85	23.9	0.00591	6.3	0.1571	24.6	1.31	37.4
Online DPM	1.5	0.55	51.2	0.00596	21.7	0.0977	53.1	0.70	66.7
	2	0.13	98.8	0.00638	57.9	0.0156	92.1	0.35	83.2
Simplified	–	0.003	99.7	0.00460	20.6	0.0150	92.5	0.31	85.1

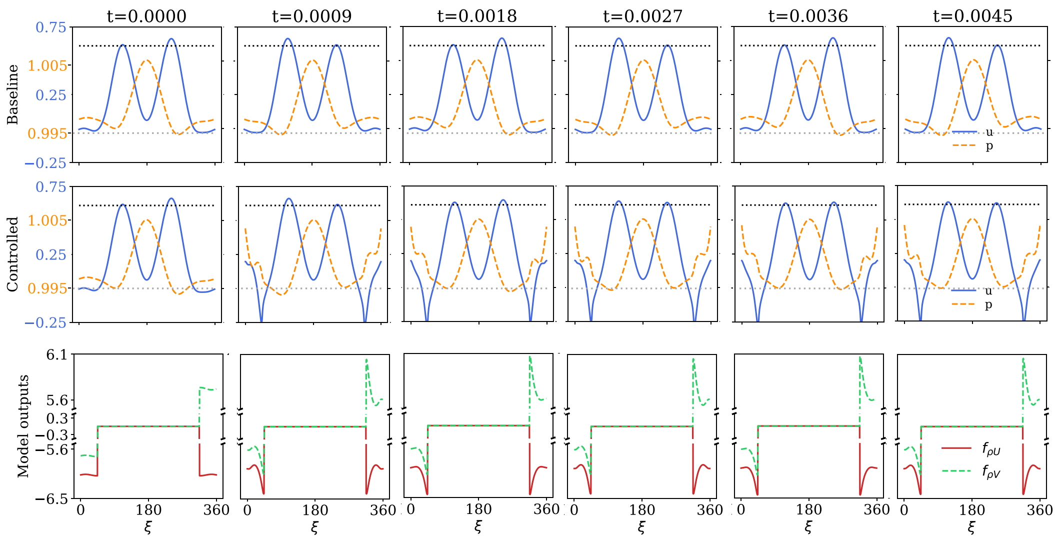

FIG. 17. Sequential snapshots of velocity (blue), pressure (orange), and control forces (red and green) along the cylinder boundary. The control-force axes are cut to show the full extents.

the observed drag reduction. The actuator forces, also shown in Fig. 17, exhibit only slight variations after $t = 6\Delta t$; this motivates the simplified (constant) actuator assessed in Sec. IV D.

F. Out-of-sample performance

The effectiveness of the one-epoch, Re $=$ 100 trained DPM controller and its constant, simplified variant are now assessed for out-of-sample Reynolds numbers between 50 and 400. Figure 18 shows the mean drag and reduction percentage for these cases. The online and simplified models both have reasonably good control performance over this range of Reynolds numbers, with the simplified model performing approximately 8% better on average (though it was not as effective at minimizing RMS quantities). The online model has out-of-sample drag reduction between 39.5% (Re $=$ 50) and 48.5% (Re $=$ 200), though its extrapolative capacity diminishes as the training-to-testing Reynolds number difference increases. This is particularly evident for Re $=$ 50, which is less than the critical Reynolds number for vortex shedding.

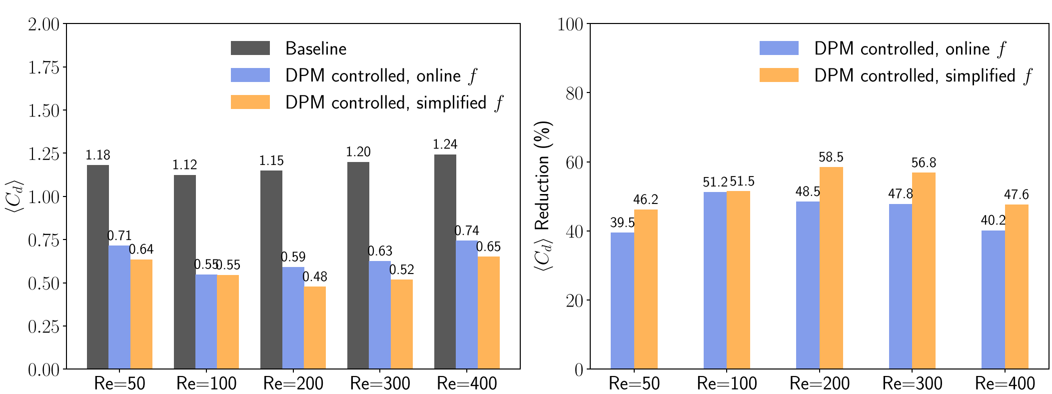

FIG. 18. Drag reduction using the online (Re $=$ 100-trained) and constant (simplified) DPM controllers for out-of-sample Reynolds numbers.

Most importantly, the DPM-trained controller is stable and relatively accurate when applied to out-of-sample Reynolds numbers. In general, the effectiveness of an active flow controller can be expected to degrade when applied far from its development regime. Here, the DPM-trained controller is successful because its training closely couples with the underlying flow physics, at least as represented by the Navier-Stokes PDEs.

V. CONCLUSION

We apply adjoint-based, PDE-constrained deep learning to develop closed-loop active controllers for flows. We assess the method's efficacy, training requirements, and out-of-sample performance, compared to benchmark *a priori* learning and exact control terms, using the method of manufactured solutions applied to the 1D viscous Burgers' equation. We then apply the method to drag reduction in a 2D incompressible Navier-Stokes flow over a confined cylinder, making efficacy and cost comparisons to deep reinforcement learning-trained control terms. Finally, we apply the method to 2D compressible Navier-Stokes flow over an unconfined cylinder and test the method's out-of-sample extrapolation to lower and higher Reynolds numbers.

The MMS viscous Burgers' example attempts to fully remove nonlinearity in the solution via control terms. These are modeled using neural networks, trained using either *a priori* learning or adjoint-based training, and they are compared to the analytic MMS control terms. For equal training loss, adjoint-based training requires 2.15% of the iterations of *a priori* training; for equal training iterations, the adjoint-based method achieves almost two orders of magnitude lower training error. The adjoint-trained controllers produce stable and accurate long-time solutions even for challenging out-of-sample control targets. The adjoint-training time window significantly influences controller effectiveness; an optimum is found between 5 and 500 time steps for this example system.

Deep reinforcement learning has recently shown promise for flow control; we compare it to adjoint-based training for drag reduction around a confined cylinder at $\mathrm{Re} = 100$. The adjoint-trained controller reduces the mean drag approximately five times as effectively as deep reinforcement learning with a neural-network controller that is several orders of magnitude less costly to train and evaluate. This difference is unsurprising, for the adjoint-based optimization fully leverages the PDE constraints during the training procedure, while the DRL-trained model relies upon an approximation to the governing equations.

The adjoint-based method is likewise successful for controlling compressible, unconfined flow over a cylinder at $\mathrm{Ma} = 0.1$. An adjoint-trained model targeting drag reduction at $\mathrm{Re} = 100$ achieves 98.8% drag reduction within two training epochs, and a simplified, constant forcing obtained from the mean outputs of the "online" controller is likewise effective. The $\mathrm{Re} = 100$-trained model is stable and successful for drag reduction between $\mathrm{Re} = 50$ and 400, with its effectiveness diminishing by less than 10% over this range.

The effectiveness of adjoint-based deep learning for laminar flow control is encouraging. Future work will focus on the model features and training methods required for deep learning control of increasingly chaotic and nonlinear flows including turbulent, high-speed, and reacting flows. These will be significantly more challenging than the laminar flows considered here due to significantly larger gradient-descent search spaces, higher training costs (especially when optimizing over three-dimensional simulations), and increasingly nonlinear and multiscale dynamics. The stability of the adjoint solution can degrade under these conditions, particularly when optimizing over longer time horizons. The use of generic body-force actuators is also limiting for applications and is presently being relaxed for turbulent flows.

ACKNOWLEDGMENTS

The authors thank J. Sirignano and T. Hickling for helpful discussions. This work was supported by the National Science Foundation under Award No. CBET-2215472. The authors acknowledge computational time on resources supported by the University of Notre Dame Center for Research Computing (CRC).

[1] M. Gad-el Hak, *Flow Control: Passive, Active, and Reactive Flow Management* (Cambridge University Press, Cambridge, 2000).

[2] M. N. Sudin, M. A. Abdullah, S. A. Shamsuddin, F. R. Ramli, and M. M. Tahir, Review of research on vehicles aerodynamic drag reduction methods, IJMEM **14**, 37 (2014).

[3] H. Wang, W. Gan, and D. Li, An investigation of the aerodynamic performance for a fuel saving double channel wing configuration, Energies **12**, 3911 (2019).

[4] L. N. Cattafesta, Q. Song, D. R. Williams, C. W. Rowley, and F. S. Alvi, Active control of flow-induced cavity oscillations, Prog. Aerosp. Sci. **44**, 479 (2008).

[5] P. Gerontakos and T. Lee, Dynamic stall flow control via a trailing-edge flap, AIAA J. **44**, 469 (2006).

[6] C. Bliamis, Z. Vlahostergios, D. Misirlis, and K. Yakinthos, Numerical Evaluation of Riblet Drag Reduction on a MALE UAV, Aerospace **9**, 218 (2022).

[7] S. Acarer, Peak lift-to-drag ratio enhancement of the DU12W262 airfoil by passive flow control and its impact on horizontal and vertical axis wind turbines, Energy **201**, 117659 (2020).

[8] D. Srinath and S. Mittal, An adjoint method for shape optimization in unsteady viscous flows, J. Comput. Phys. **229**, 1994 (2010).

[9] L. Prandtl, Über Flüssigkeitsbewegung bei sehr kleiner Reibung, in *Verhandl. 3rd Int. Math. Kongr. Heidelberg (1904)*, Heidelberg, Germany (1905), pp. 484–491.

[10] J. Rabault, M. Kuchta, A. Jensen, U. Réglade, and N. Cerardi, Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control, J. Fluid Mech. **865**, 281 (2019).

[11] H. Tang, J. Rabault, A. Kuhnle, Y. Wang, and T. Wang, Robust active flow control over a range of Reynolds numbers using an artificial neural network trained through deep reinforcement learning, Phys. Fluids **32**, 053605 (2020).

[12] K. Roussopoulos, Feedback control of vortex shedding at low reynolds numbers, J. Fluid Mech. **248**, 267 (1993).

[13] H. Choi, R. Temam, P. Moin, and J. Kim, Feedback control for unsteady flow and its application to the stochastic Burgers equation, J. Fluid Mech. **253**, 509 (1993).

[14] S. L. Brunton and B. R. Noack, Closed-loop turbulence control: Progress and challenges, Appl. Mech. Rev. **67**, 050801 (2015).

[15] G. Farrell, M. Gibbons, and T. Persoons, Combined passive/active flow control of drag and lift forces on a cylinder in crossflow using a synthetic jet actuator and porous coatings, Actuators **11**, 201 (2022).

[16] S. S. Collis, R. D. Joslin, A. Seifert, and V. Theofilis, Issues in active flow control: theory, control, simulation, and experiment, Prog. Aerosp. Sci. **40**, 237 (2004).

[17] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, Machine learning for fluid mechanics, Annu. Rev. Fluid Mech. **52**, 477 (2020).

[18] K. Portal-Porras, U. Fernandez-Gamiz, E. Zulueta, A. Ballesteros-Coll, and A. Zulueta, CNN-based flow control device modelling on aerodynamic airfoils, Sci. Rep. **12**, 8205 (2022).

[19] P. Liao, W. Song, P. Du, and H. Zhao, Multi-fidelity convolutional neural network surrogate model for aerodynamic optimization based on transfer learning, Phys. Fluids **33**, 127121 (2021).

[20] A. T. Mohan and D. V. Gaitonde, A deep learning based approach to reduced order modeling for turbulent flow control using LSTM neural networks, arXiv:1804.09269.

[21] S. E. Otto and C. W. Rowley, Linearly recurrent autoencoder networks for learning dynamics, SIAM J. Appl. Dyn. Syst. **18**, 558 (2019).

[22] J. Viquerat, J. Rabault, A. Kuhnle, H. Ghraieb, A. Larcher, and E. Hachem, Direct shape optimization through deep reinforcement learning, J. Comput. Phys. **428**, 110080 (2021).

[23] T. Sonoda, Z. Liu, T. Itoh, and Y. Hasegawa, Reinforcement learning of control strategies for reducing skin friction drag in a fully developed turbulent channel flow, J. Fluid Mech. **960**, A30 (2023).
[24] T. Lee, J. Kim, and C. Lee, Turbulence control for drag reduction through deep reinforcement learning, Phys. Rev. Fluids **8**, 024604 (2023).
[25] R. Paris, S. Beneddine, and J. Dandois, Robust flow control and optimal sensor placement using deep reinforcement learning, J. Fluid Mech. **913**, A25 (2021).
[26] J. Li and M. Zhang, Reinforcement-learning-based control of confined cylinder wakes with stability analyses, J. Fluid Mech. **932**, A44 (2022).
[27] F. Pino, L. Schena, J. Rabault, and M. A. Mendez, Comparative analysis of machine learning methods for active flow control, J. Fluid Mech. **958**, A39 (2023).
[28] C. Vignon, J. Rabault, and R. Vinuesa, Recent advances in applying deep reinforcement learning for flow control: Perspectives and future directions, Phys. Fluids **35**, 031301 (2023).
[29] Y. Li, Deep reinforcement learning: An overview, arXiv:1701.07274.
[30] D. Andre and S. J. Russell, State abstraction for programmable reinforcement learning agents, in AAAI-02 Proceedings (AAAI, Alberta, Canada, 2002), pp. 119–125.
[31] D. Abel, D. Arumugam, L. Lehnert, and M. Littman, State abstractions for lifelong reinforcement learning, in *International Conference on Machine Learning* (PMLR, Stockholmsmässan, Stockholm Sweden, 2018), pp. 10–19.
[32] Q. Wang, L. Yan, G. Hu, W. Chen, J. Rabault, and B. R. Noack, Dynamic feature-based deep reinforcement learning for flow control of circular cylinder with sparse surface pressure sensing, arXiv:2307.01995.
[33] A. Carnarius, F. Thiele, E. Özkaya, A. Nemili, and N. R. Gauger, Optimal control of unsteady flows using a discrete and a continuous adjoint approach, in *System Modeling and Optimization: 25th IFIP TC 7 Conference, CSMO 2011, Berlin, 2011, Revised Selected Papers 25* (Springer, Berlin, Germany, 2013), pp. 318–327.
[34] R. Bellman, Dynamic programming and lagrange multipliers, Proc. Natl. Acad. Sci. (USA) **42**, 767 (1956).
[35] P. Tsiotras and M. Mesbahi, Toward an algorithmic control theory, J. Guid. Control. Dyn. **40**, 194 (2017).
[36] A. Jameson, Aerodynamic design via control theory, J. Sci. Comput. **3**, 233 (1988).
[37] J. J. Reuther, A. Jameson, J. J. Alonso, M. J. Rimllnger, and D. Saunders, Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 2, J. Aircr. **36**, 61 (1999).
[38] A. A. Gorodetsky and J. D. Jakeman, Gradient-based optimization for regression in the functional tensor-train format, J. Comput. Phys. **374**, 1219 (2018).
[39] A. Carnarius, F. Thiele, E. Özkaya, and N. R. Gauger, Adjoint approaches for optimal flow control, in *5th Flow Control Conference, Chicago* (2010), p. 5088.
[40] J. W. He, R. Glowinski, R. Metcalfe, A. Nordlander, and J. Periaux, Active control and drag optimization for flow past a circular cylinder: I. oscillatory cylinder rotation, J. Comput. Phys. **163**, 83 (2000).
[41] A. Kord and J. Capecelatro, Optimal perturbations for controlling the growth of a Rayleigh-Taylor instability, J. Fluid Mech. **876**, 150 (2019).
[42] A. Marta and J. Alonso, High-speed MHD flow control using adjoint-based sensitivities, in *14th AIAA/AHI Space Planes and Hypersonic Systems and Technologies Conference* (AIAA, Canberra, Australia, 2006), p. 8009.
[43] J. Kim and T. R. Bewley, A linear systems approach to flow control, Annu. Rev. Fluid Mech. **39**, 383 (2007).
[44] J. Sirignano, J. F. MacArt, and J. B. Freund, DPM: A deep learning PDE augmentation method with application to large-eddy simulation, J. Comput. Phys. **423**, 109811 (2020).
[45] J. F. MacArt, J. Sirignano, and J. B. Freund, Embedded training of neural-network subgrid-scale turbulence models, Phys. Rev. Fluids **6**, 050502 (2021).
[46] J. Sirignano, J. MacArt, and K. Spiliopoulos, PDE-constrained models with neural network terms: Optimization and global convergence, J. Comput. Phys. **481**, 112016 (2023).
[47] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. (MIT Press, Cambridge, MA, 2018).

[48] V. Konda and J. Tsitsiklis, Actor-critic algorithms, in *Adv. Neural Inf. Process. Syst.*, edited by S. Solla, T. Leen, and K. Müller (MIT Press, 1999), Vol. 12.

[49] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal policy optimization algorithms, arXiv:1707.06347.

[50] A. Kuhnle, M. Schaarschmidt, and K. Fricke, Tensorforce: A TensorFlow library for applied reinforcement learning, Web page (2017), https://github.com/tensorforce/tensorforce.

[51] 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 *Adv. Neural Inf. Process. Syst.*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019), Vol. 32, pp. 8024–8035.

[52] G. B. Whitham, *Linear and Nonlinear Waves* (John Wiley & Sons, New York, 2011).

[53] N. Geneva and N. Zabaras, Modeling the dynamics of PDE systems with physics-constrained deep autoregressive networks, J. Comput. Phys. **403**, 109056 (2020).

[54] R. Vishnampet Ganapathi Subramanian, An exact and consistent adjoint method for high-fidelity discretization of the compressible flow equations, Ph.D. thesis, UIUC, 2015.

[55] J. Rabault and A. Kuhnle, Accelerating deep reinforcement learning strategies of flow control through a multi-environment approach, Phys. Fluids **31**, 094105 (2019).

[56] M. Schäfer, S. Turek, F. Durst, E. Krause, and R. Rannacher, *Benchmark Computations of Laminar Flow Around a Cylinder* (Springer, New York, 1996).

[57] *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS book*, edited by A. Logg, K.-A. Mardal, and G. N. Wells (Springer Science & Business Media, New York, 2012), Vol. 84.

[58] S. K. Mitusch, S. W. Funke, and M. Kuchta, Hybrid FEM-NN models: Combining artificial neural networks with the finite element method, J. Comput. Phys. **446**, 110651 (2021).

[59] S. K. Mitusch, S. W. Funke, and J. S. Dokken, Dolfin-Adjoint: Automated Adjoints for FEniCS and Firedrake, J. Open Source Softw. **4**, 1292 (2019).

[60] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization, ACM Trans. Math. Softw. **23**, 550 (1997).

[61] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, TensorFlow: A system for large-scale machine learning, in *12th OSDI'16, Savannah, (Savannah, GA)* (USENIX Association, 2016), pp. 265–283.

[62] S. K. Lele, Compact finite difference schemes with spectral-like resolution, J. Comput. Phys. **103**, 16 (1992).

[63] M. M. Zdravkovich, *Flow Around Circular Cylinders: Applications* (Oxford University Press, Oxford, UK, 1997), Vol. 2.

[64] B. Rajani, A. Kandasamy, and S. Majumdar, Numerical simulation of laminar flow past a circular cylinder, Appl. Math. Modell. **33**, 1228 (2009).

[65] S. Sen, S. Mittal, and G. Biswas, Steady separated flow past a circular cylinder at low Reynolds numbers, J. Fluid Mech. **620**, 89 (2009).

[66] C. Norberg, Pressure forces on a circular cylinder in cross flow, in *Bluff-Body Wakes, Dynamics and Instabilities* (Springer, New York, 1993), pp. 275–278.

[67] C. H. K. Williamson and A. Roshko, Measurements of base pressure in the wake of a cylinder at low Reynolds numbers, Z. Flugwissensch. **14**, 38 (1990).