
NeuralFluid : Nueral Fluidic System Design and Control with Differentiable Simulation

Yifei Li
MIT CSAIL

Yuchen Sun
Georgia Institute of Technology

Pingchuan Ma
MIT CSAIL

Eftychios Sifakis
University of Wisconsin-Madison

Tao Du
Tsinghua University, Shanghai Qi Zhi Institute

Bo Zhu
Georgia Institute of Technology

Wojciech Matusik
MIT CSAIL

Abstract

We present ***NeuralFluid***, a novel framework to explore neural control and design of complex fluidic systems with dynamic solid boundaries. Our system features a fast differentiable Navier-Stokes solver with solid-fluid interface handling, a low-dimensional differentiable parametric geometry representation, a control-shape co-design algorithm, and gym-like simulation environments to facilitate various fluidic control design applications. Additionally, we present a benchmark of design, control, and learning tasks on high-fidelity, high-resolution dynamic fluid environments that pose challenges for existing differentiable fluid simulators. These tasks include designing the control of artificial hearts, identifying robotic end-effector shapes, and controlling a fluid gate. By seamlessly incorporating our differentiable fluid simulator into a learning framework, we demonstrate successful design, control, and learning results that surpass gradient-free solutions in these benchmark tasks.

1 Introduction

Complex fluidic systems play an important role in many engineering and scientific disciplines, encompassing applications at different length scales ranging from biomedical implants [1], microfluidic devices [2], hydraulic devices to and flying robots [3]. Understanding these fluid-solid coupling mechanisms in nature and mimicking their control strategies in artificial designs is essential for advancing our control and design capabilities to synthesize novel solid-fluid systems.

Devising neural control algorithms to accurately manipulate the behavior of a complex fluidic system and optimize its performance remains challenging due to the intricate interplay between device geometry, control policies, flow dynamics, and the inherent physical and optimization constraints unique to each fluidic system. On one hand, differentiable simulation fluid-system interactions are inherently difficult because simulation is dynamic, involving a sequence of forward and backward steps interleaved with control signals that are computationally expensive. On the other hand, naively employing traditional control algorithms, mainly derived from their solid counterparts, to control fluidic systems remains difficult due to characterizing the infinite degrees of freedom of fluid flows and their interactions with solid boundaries. The co-design of fluid-solid systems, involving both shape and control, is critical to exploring the optimal performance of these systems.

Currently, the machine learning community lacks a computational Gym-like [4] environment to facilitate the exploration of fluidic systems manifesting strong solid-fluid interactions and controllable

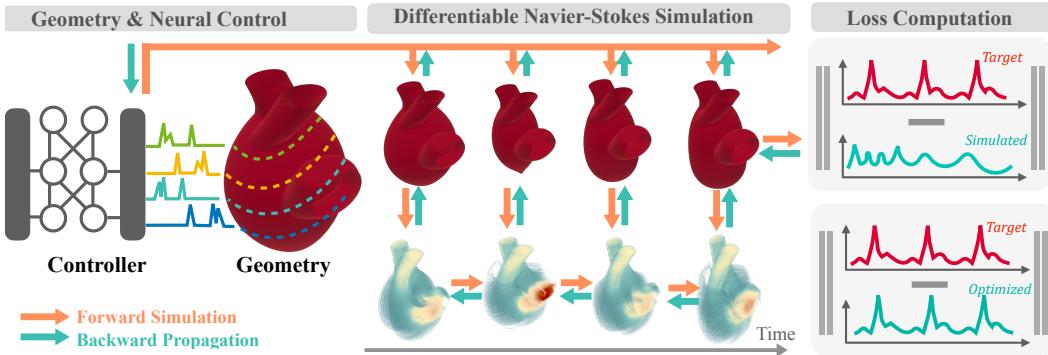


Figure 1: **Pipeline Overview.** (1) Our pipeline starts with an initial parametric geometry and a neural network parameterized controller. (2) The fluid dynamics is then simulated using a dynamic Navier-Stokes solver. (3) The performance of the design and control is evaluated using a loss function, the gradients of which are then back-propagated through our end-to-end differentiable framework. (4) The gradient-based optimization iteratively improves the geometry and control to achieve the task goal. This pipeline allows for efficient geometry and control co-optimization.

dynamic boundaries. Recent literature in robotic learning (e.g., [5]) has established unified multi-physics differentiable simulation platforms to facilitate learning control policies for various fluid interactions in daily scenarios. Similar ideas can be observed in [6, 7, 8], where differentiable simulation plays a central role in accommodating various design and optimization tasks of dynamic systems involving fluid dynamics. However, despite these inspiring advances, learning the control policies and exploring the optimal performance of a dynamic fluidic system with complex boundary conditions remains difficult due to their inherent complexities in differentiating solid boundary behaviors and optimizing their fluidic consequences due to these boundary motions.

This paper presents a novel framework for a fully automated pipeline aimed at devising neural controls for complex fluidic systems with dynamic boundaries. Our framework is designed to robustly control complex fluidic systems that consist of externally driven soft boundaries and internal complex flow behaviors, such as those systems underpinning an artificial heart or a microfluidic device.

NeuralFluid consists of three critical components to enable neural control of a complex fluidic system. First, we devise a differentiable geometry representation to offer an expressive design space while remaining low-dimensional, enabling efficient exploration by the optimization algorithm. Second, we implement a differentiable fluid simulator with solid-fluid interface handling to accurately characterize the dynamic fluid behavior and predict its spatiotemporal impact on the moving boundaries. We back-propagate gradients at the solid-fluid interface to extend gradient computation to the geometry iso-surface. Last, we provided an optimization framework to efficiently search the design space, considering the underlying fluid dynamics and boundary conditions.

Our pipeline features a low-dimensional parametric geometry representation capable of expressing complex shapes and a differentiable Navier-Stokes simulator with geometry gradient computation for predicting dynamic fluid behavior in response to control signals. In addition, our pipeline leverages gradient-based optimization for efficient design space exploration, co-optimization of the device geometry and control, and accurate performance evaluation of the design under dynamic flows. To showcase the practical implications and versatility of our approach, we have established a suite of Gym-like [4] environments. These benchmarks are designed to test applications in robotics and engineering, facilitating advancements in system identification, optimization of end-effector shapes and controls, and the dynamic optimization of structures such as artificial hearts within a closed-loop control framework. We showcase the effectiveness of our pipeline in facilitating different design and control tasks, including amplifier, fluidic switch, flow modulator, shape and position identification, closed-loop control of water gate and artificial heart.

We summarize our main contributions as follows:

- Development of a fast differentiable Navier-Stokes simulator for optimization in 2D and 3D scenes.

- Development of a low-dimensional differentiable parametric geometry representation for complex shapes embedded into the differentiable simulation pipeline.
- Gradient computation extension to geometry iso-surface to enable control and geometry co-design and iso-surface optimization.
- Gym-like [4] environments and benchmarks to demonstrate applications in robotics and engineering, including the design of amplifier, fluidic switch, flow modulator, geometry system identification, and closed-loop control of a fluid gate and artificial heart.

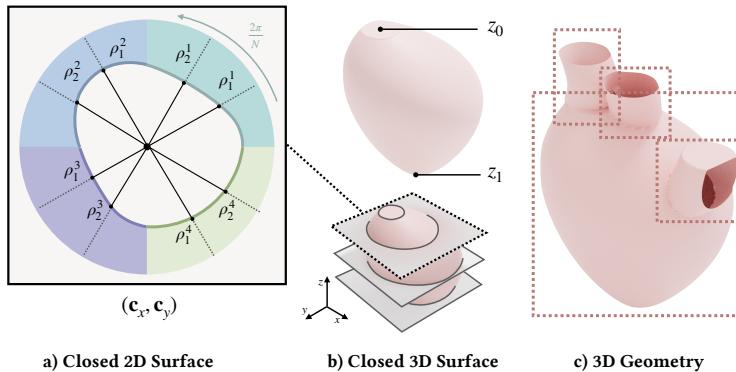
2 Method

2.1 Pipeline Overview

We present an overview of our method in Fig. 1. Our pipeline defines the designs with a low-dimensional parametric geometry representation (Sec. 2.2). The behavior and performance of the design in the fluid environment are evaluated by a dynamic differentiable Navier-Stokes simulator (Sec. 2.3). Both components are embedded in a gradient-based optimization framework that co-optimizes both the geometric design and the control signal until convergence.

2.2 Geometry Representation

We represent our geometry with a low-dimension representation. Take the illustration in the inset figure as an example, here we introduce the representation on a high level, and refer the readers to the appendix for the full details. We parameterize a closed 2D surface using its center \mathbf{c} and a set of connected Bezier curves with their control points de-



fine in polar coordinates ρ_i for $i \in [1, 2, \dots, 2N]$, where every two control points define a 2D Bezier curve spanning $\frac{2\pi}{N}$ radians in the polar coordinate system. This representation offers a compact way of defining diverse geometries. We further parameterize a closed 3D surface using a list of 2D surfaces defining the key cross-sections of the geometry along an extrusion axis z of the local object frame, where each 2D surface is parameterized as described above. The parameterization includes $z = z_0$ and z_1 , which determines the Z plane of the first and last cross-section, along with the parameters for each key 2D cross-section, which are assumed to be evenly spaced between $z \in [z_0, z_1]$. The continuous geometry interpolates the key cross-sections along the z-axis. Finally, we construct more complex 3D geometries using operations from Constructive Solid Geometry (CSG): Union and intersection, which allows us to define a 3D parametric heart model using the union of four sub-geometries.

2.3 Differentiable Navier-Stokes Simulation

Our fluid dynamics is governed by the incompressible Navier-Stokes equations. These consist of the momentum equation (Eq. 1a), accounting for temporal changes in velocity (\mathbf{u}), advective acceleration, viscous dissipation, and pressure (p) gradient forces for an incompressible fluid with fluid density ρ and kinematic viscosity ν . The incompressibility condition (Eq. 1b) requires the divergence of the velocity field must be zero to enforce the conservation of mass:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla) \mathbf{u} + \nu \nabla^2 \mathbf{u} - \frac{1}{\rho} \nabla p, \\ \nabla \cdot \mathbf{u} = 0 \end{cases} \quad (1a) \quad (1b)$$

2.3.1 Numerical Simulation

We build the fluid simulator by leveraging the operator-splitting method [9][10]. A single simulation step comprises three sub-steps: advection, viscosity, and projection. See the Appendix B for details on time discretization. The simulation domain is discretized on a standard Marker-and-Cell (MAC) grid [11], with pressures stored at cell centers and velocities at cell faces. By employing the finite-difference scheme on the MAC grid cells and faces, we construct the matrix $\frac{1}{\Delta x} \mathbf{G}$ for gradient operator and its negative transpose $-\frac{1}{\Delta x} \mathbf{G}^T$ for divergence operator. In the following sections, capital letters will refer to matrices or the flattened vectors induced by the fields denoted by the corresponding lowercase letters in Appendix B.

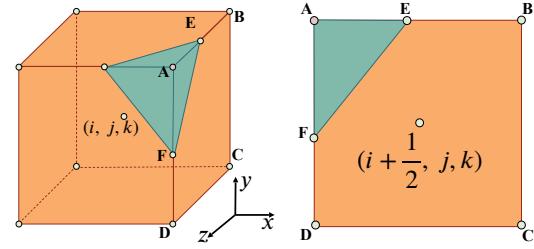
Advection We employ the semi-Lagrangian advection scheme, where the advected velocity field $\tilde{\mathbf{U}}^{n+1}$ is a linear interpolation of the velocity field \mathbf{U}^n . The interpolation position function is a function of \mathbf{U}^n can be put into a matrix form \mathbf{B} , which results in:

$$\tilde{\mathbf{U}}^{n+1} = \mathbf{B}(\mathbf{U}^n) \mathbf{U}^n. \quad (2)$$

Viscosity For incompressible fluid with a constant viscosity coefficient, the viscous force density is the product of the Laplacian of velocity and the viscosity coefficient. For each axis, the Laplacian of the corresponding velocity component is calculated on grid points using the finite difference method:

$$\hat{\mathbf{U}}^{n+1} = \left(\mathbf{I} - \frac{\nu \Delta t}{\Delta x^2} \mathbf{G}^T \mathbf{G} \right) \tilde{\mathbf{U}}^{n+1}. \quad (3)$$

Projection The projection step ensures the incompressibility of the fluid. Solid unaligned with the grid may intersect with grid faces, which can be captured with a cut-cell method [12]. We introduce α^{n+1} to represent the fluid proportion of a grid face. The solid's signed distance function (SDF) ϕ^{n+1} and the velocity \mathbf{u}_s^{n+1} can be derived from the solid geometry. We first use marching cube to compute the geometry zero contour. Next, we identify the intersection points between this contour and the grid faces and compute α^{n+1} based on ϕ . For instance, in the inset figure, grid face $(i + \frac{1}{2}, j, k)$ is cut by the contour, then $\alpha_{i+\frac{1}{2}, j, k}^{n+1} = \frac{S_{AEF}}{S_{ABCD}} = \frac{1}{2} \cdot \frac{|AE|}{|AB|} \cdot \frac{|AF|}{|AD|} = \frac{1}{2} \cdot \frac{\phi_A^{n+1}}{\phi_A^{n+1} - \phi_B^{n+1}} \cdot \frac{\phi_A^{n+1}}{\phi_A^{n+1} - \phi_D^{n+1}}$.



The volume change rates for fluid and solid at grid cell (i, j, k) , denoted as $\gamma_{f,i,j,k}^{n+1}$ and $\gamma_{s,i,j,k}^{n+1}$ respectively, equal to the sum of flux on the cell's surrounding faces, which can be calculated using α^{n+1} , fluid velocity \mathbf{u}^{n+1} , and solid velocity \mathbf{u}_s^{n+1} .

The incompressibility condition gives requires the sum of $\gamma_{f,i,j,k}^{n+1}$ and $\gamma_{s,i,j,k}^{n+1}$ to be zero, which gives

$$\frac{\Delta t}{\rho \Delta x} \mathbf{G}^T \mathbf{S}^{n+1} \mathbf{G} \mathbf{P}^{n+1} = \mathbf{G}^T \mathbf{S}^{n+1} \hat{\mathbf{U}}^{n+1} + \mathbf{G}^T (\mathbf{I} - \mathbf{S}^{n+1}) \mathbf{U}_s^{n+1}. \quad (4)$$

where \mathbf{S}^{n+1} is a diagonal matrix induced by α^{n+1} , and \mathbf{P} is the pressure. After solving the linear system, the fluid velocity is updated based on the pressure values:

$$\mathbf{U}^{n+1} = \hat{\mathbf{U}}^{n+1} - \frac{\Delta t}{\rho \Delta x} \mathbf{G} \mathbf{P}^{n+1}, \quad (5)$$

2.3.2 Back-propagation through Time

We construct our back-propagation algorithm to mirror the sequence of operations carried out in the forward pass but in a reversed order.

Given the gradients of the loss function J with respect to the velocity field \mathbf{u} at time step $n + 1$, denoted by $\frac{\partial J}{\partial \mathbf{U}^{n+1}}$, our goal is to compute the corresponding gradients at time step n , $\frac{\partial J}{\partial \mathbf{U}^n}$.

Projection We begin by reversing the projection step to back-propagate $\frac{\partial J}{\partial \mathbf{U}^{n+1}}$ to derive $\frac{\partial J}{\partial \mathbf{P}^{n+1}}$ and $\frac{\partial J}{\partial \hat{\mathbf{U}}^{n+1}}$. Back-propagating through Eq. 5 gives

$$\frac{\partial J}{\partial \mathbf{P}^{n+1}} = -\frac{\Delta t}{\rho \Delta x} \frac{\partial J}{\partial \mathbf{U}^{n+1}} \mathbf{G}. \quad (6)$$

We can back-propagate the adjoint of Eq. 4 w.r.t $\hat{\mathbf{U}}^{n+1}$ by defining the adjoint variable \mathbf{y} and derive

$$\frac{\partial J}{\partial \hat{\mathbf{U}}^{n+1}} = \frac{\partial J}{\partial \mathbf{U}^{n+1}} + \mathbf{y} \mathbf{G}^T \mathbf{S}^{n+1}, \quad (7)$$

where $\mathbf{A} = \frac{\Delta t}{\rho \Delta x} \mathbf{G}^T \mathbf{S}^{n+1} \mathbf{G}$, $\mathbf{b} = \mathbf{G}^T \mathbf{S}^{n+1} \hat{\mathbf{U}}^{n+1} + \mathbf{G}^T (\mathbf{I} - \mathbf{S}^{n+1}) \mathbf{U}_s^{n+1}$, and \mathbf{y} is computed by solving the linear system $\mathbf{A} \mathbf{y}^T = (\frac{\partial J}{\partial \mathbf{P}^{n+1}})^T$.

Viscosity and Advection Back-propagating through viscosity and advection simply involves back-propagating $\frac{\partial J}{\partial \hat{\mathbf{U}}^{n+1}}$ through Eq. 3 and Eq. 2, which allows us to derive:

$$\frac{\partial J}{\partial \mathbf{U}^n} = \frac{\partial J}{\partial \hat{\mathbf{U}}^{n+1}} \left(\mathbf{I} - \frac{\nu \Delta t}{\Delta x^2} \mathbf{G}^T \mathbf{G} \right) \cdot \left(\frac{\partial \mathbf{B}}{\partial \mathbf{U}^n} \mathbf{U}^n + \mathbf{B}(\mathbf{U}^n) \right). \quad (8)$$

The above equations provide the outline of the back-propagation process through a single time step of from time step $n+1$ to n . To compute the gradients of the loss function J at any time step, we iterate the back-propagation process over the full sequence of time steps.

2.3.3 Back-propagation through Geometry

The parametric geometry affects simulation through the solid-fluid boundary during the projection step in Eq. 4. Specifically, the SDF of the geometry ϕ^{n+1} affects the volume matrix \mathbf{S}^{n+1} and the velocity (in the case of moving geometry) of the geometry \mathbf{U}_s^{n+1} affects the boundary condition. We can back-propagate $\frac{\partial J}{\partial \mathbf{P}^{n+1}}$ w.r.t these two parameters to derive

$$\left\{ \begin{array}{l} \frac{\partial J}{\partial \mathbf{S}^{n+1}} = \mathbf{y} \mathbf{G}^T (\hat{\mathbf{U}}^{n+1} - \mathbf{U}_s^{n+1}) + \mathbf{y} \frac{\partial \mathbf{A}}{\partial \mathbf{S}^{n+1}} \mathbf{P}^{n+1}, \end{array} \right. \quad (9a)$$

$$\left\{ \begin{array}{l} \frac{\partial J}{\partial \mathbf{U}_s^{n+1}} = -\mathbf{y} \mathbf{G}^T \mathbf{S}^{n+1}. \end{array} \right. \quad (9b)$$

Further back-propagating $\frac{\partial J}{\partial \mathbf{S}^{n+1}}$ first through the SDF ϕ then through the distance computation and $\frac{\partial J}{\partial \mathbf{U}_s^{n+1}}$ through geometry velocity function allows us to optimize through the geometry iso-surface.

2.3.4 Neural Fluid Control

We can train neural-network parameterized closed-loop fluid controllers with gradients fully computed at both geometry and velocity throughout time. We parameterize our controllers with a two-layer MLP. The controller takes as input the observation of the fluid velocity field at each frame and outputs dynamic control signals that affect the geometry through our parametric geometry presentation, which further affects the flow field. Our fully differentiable framework allows gradient-based methods to train the controller efficiently. We implemented the backbone of our code in C++ and CUDA for computational efficiency. We derived gradients for the geometry and simulation module analytically, then exposed the differentiable simulation framework through pybind11 [13] to enable seamless integration with deep learning libraries, which in our case is PyTorch [14].

3 Benchmarks and Applications

In this section, we introduce our fluidic design benchmarks and environments. We warp our environments using the standard protocol in the gym to facilitate learning practices. We present six fluidic design and control tasks (Fig. 2) to assess the effectiveness of our computational pipeline for fluidic

Table 1: **Task Specifications.** We summarize the simulation and optimization configuration for the design tasks shown in Sec. 3 and report the initial and optimized loss. We note that because our implementation adopts CFL condition for numerical stability during simulation, the actual steps simulated and back-propagated are higher than the numbers shown in “# Frames”.

	Resolution	# Frames	# Param.	$\frac{\partial L_f}{\partial \text{Design}}$	$\frac{\partial L_f}{\partial \text{Control}}$	Loss (L_f)	Initial	Optimized
Amplifier	64×64	40	5	✓		13.401	0.005	
Switch	64×64	120	10	✓	✓	13.162	1.893	
Shape Identifier	128×128	10	10	✓		70.152	0.759	
Flow Modulator	$40 \times 40 \times 40$	100	34	✓	✓	6.118	0.069	
Neural Gate	$40 \times 40 \times 40$	50	4.5k		✓	7.318	0.000	
Neural Heart	$48 \times 48 \times 48$	180	7.1k	✓	✓	1.086	0.004	

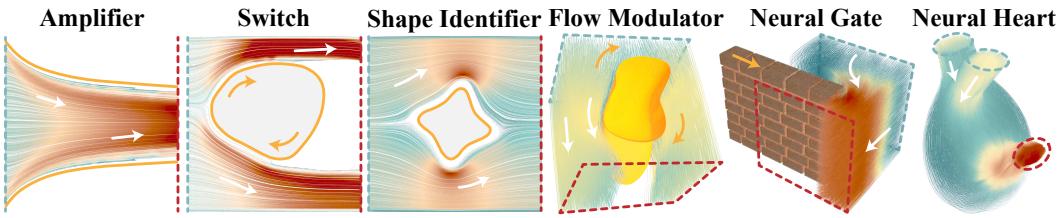


Figure 2: **Tasks Overview.** In each task, the blue dashed line represents the inlet, the red dashed line indicates the outlet, the white arrows show the flow direction, and the orange shapes and arrows denote the geometry and its motion direction.

system design and learning. A comprehensive illustration of these design scenarios, including the visualization of the optimization process, is provided in Appendix Sec C and our supplemental video. Initial conditions are set for all optimizations using randomly sampled values. We use Adam as our optimizer. We summarize the simulation configuration and optimization configuration as well as relevant statistics in Table 1.

3.1 Task Overview

Amplifier This design problem aims to amplify a parallel horizontal inflow by three times from an initial velocity of 5 units to 15 units. The device boundaries are parameterized as two symmetrically placed cubic Bezier curves. The design variables are the control points and endpoints of the two curves. The loss function is defined as the last frame L2 norm of the difference between the target and optimized fluid velocity norm. We visualize the initial and optimized designs in Fig. 3a. We overlay the design and the corresponding velocity field (colored by the norm) for both iterations.

Shape Identifier This task provides an example of system identification in a fluid environment by identifying the shape and position of a geometry, given observations of the flow field. We randomly

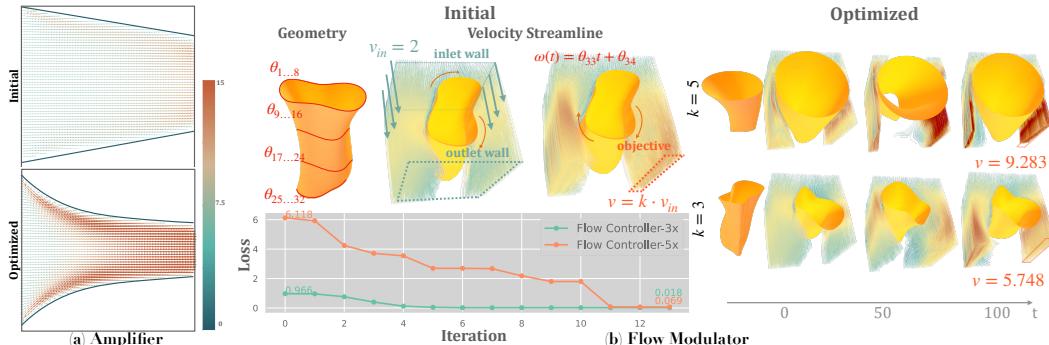


Figure 3: (a) Visualization of *Amplifier*. (b) Visualization of *Flow Modulator*.

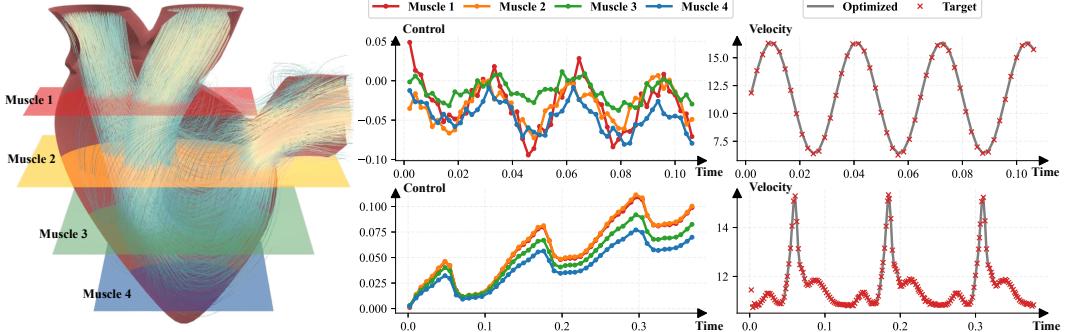


Figure 4: **Artificial Heart.** *Left:* visualization of the domain and the location of the muscles. *Middle:* Optimized control policy rollout visualization. *Right:* Optimization results visualization. The top and bottom diagrams visualize the cosine and the ECG target variants.

initialize the geometry in the domain. We define the loss function as the sum of the L2 norm of the velocity field difference to the observed ground-truth flow field across time. The optimization successfully reconstructs the shape and its position with random initialization (See Exp. 4.2).

Switch This task simultaneously optimizes the geometry and the constant rotational speed of a 2D switch, allowing dynamic regulation of outlet flow velocity. A horizontal inflow at the left interacts with the switch, splitting into two distinct streams towards the right side. The goal is to let the time-dependent average velocity norm of the upper stream align with a predetermined flow profile. We define the loss function as the sum of the L2 norm of the difference between the average velocity norm of the fluid and the target velocity at each frame. The optimized design and control successfully generate a linearly increasing upper stream velocity norm profile, matching the specified target.

Flow Modulator This task optimizes the geometry and control of a rotating 3D flow controller to achieve a target average outlet flow ($k \times$ inflow) at the domain’s right boundary by the end of the simulation. The controller’s geometry is parameterized by four 3D cross-sections, with rotation controlled by a sinusoidal function. We define the loss as the L2 norm of the difference between the average outlet and target velocity at the final frame. Fig. 3b illustrates the task: the top left shows initial parameters and task specification, bottom left shows optimization trajectories, and the right visualizes the optimized geometry and velocity streamline with two variants ($k = 3, 5$).

Neural Gate Controller We learn a closed-loop controller for a 3D fluid gate moving horizontally. The controller is parameterized as a two-layer MLP. It observes the current outflow velocity through the gate and outputs the next frame motion offset to control the outflow velocity to match a target.

3.2 Scalability to Complex Fluid Fields: A Case Study of Artificial Heart

We showcase the scalability of our method through an artificial heart design and control learning task (Fig. 4). Artificial heart development is difficult due to the complex blood flow movement within the heart. This case study provides a first step in studying the heart’s control strategies. We train a closed-loop controller that outputs the per-time-step contraction signal of the four muscles of a simplified heart model so that the outlet velocity matches a pre-defined target profile. The controller’s states include temporal encoding of the current time step and the current outflow norm, and they are parameterized using a two-layer MLP. The heart’s geometry is parameterized as the union of the two inlets, one outlet, and the heart chamber. In two variants of the task, one target flow profile is parameterized using a cosine curve (Fig. 4 top), and one target flow profile mimics the shape of an electrocardiogram (Fig. 4 bottom). We define the loss function as the sum of the L2 norm of the difference between the average velocity norm of the fluid and the target velocity at each frame. In both variants, the trained controller successfully outputs signals that generate blood flow that matches the target, demonstrating the effectiveness of our gradient-based optimization framework. We further visualize the rollouts of the trained controllers at Fig 4 left.

Table 2: **Time Performance.** Our method achieves one order of magnitude speedup across all resolutions compared to PhiFlow in both forward simulation and backward gradient propagation.

Resolution	Forward			Backward		
	PhiFlow (s)	Ours (s)	Speedup	PhiFlow (s)	Ours (s)	Speedup
$32 \times 32 \times 32$	1.282	0.024	53.4 ×	1.546	0.095	16.3 ×
$40 \times 40 \times 40$	1.741	0.039	44.6 ×	1.983	0.121	16.4 ×
$48 \times 48 \times 48$	2.227	0.068	32.8 ×	2.412	0.158	15.3 ×
$64 \times 64 \times 64$	3.145	0.105	30.0 ×	4.094	0.301	13.6 ×

Table 3: **Memory and time performance comparison with DiffTaichi**

Resolution	Memory (MB)		Forward Time (s)		Backward Time (s)	
	DiffTaichi	Ours	DiffTaichi	Ours	DiffTaichi	Ours
$32 \times 32 \times 32$	685	292	0.081	0.024	0.074	0.027
$40 \times 40 \times 40$	1136	308	0.146	0.039	0.133	0.041
$48 \times 48 \times 48$	1805	322	0.228	0.068	0.183	0.064
$64 \times 64 \times 64$	5005	405	0.435	0.105	0.363	0.117

4 Experiments

4.1 Effects of Initialization on Optimization

This experiment studies the effect of random initialization in our fluid optimization tasks. Specifically, we conduct an experiment on the 3D heart controller task, which utilizes a neural network with 7,100 parameters. For this task, we initialized the network parameters with five different random seeds, tracking convergence under each condition. In the accompanying figure, we plot an extended version of the training trajectory, scaled logarithmically for better visualization, to compare the convergence of different random initializations. However, in practice, our method achieves the objective driven by the loss within tens of iterations, as is evident from the steep initial descent in the optimization curve. As shown in Fig. 5 left, the optimization consistently converges across all seeds, despite variations in initial network behavior. This consistency indicates the robustness of our method to random initialization even in high-dimensional optimization spaces, supporting its application to complex tasks in differentiable physics. These curves offer valuable guidelines for practitioners using our method in their deployments: the gradients from our approach are robust to hyper-parameters and scalable to high-dimensional optimization problems.

4.2 Gradient-Based vs Gradient-Free Optimization

We study the effectiveness of our gradient-based method against gradient-free optimization methods in the *Neural Heart* task (Fig. 5 right). We choose Proximal Policy Optimization (PPO) [15] as the baseline for reinforcement learning and CMA-ES [16] for evolution strategies. This environment is particular challenging due to the sensitivity of the bloodflow to the control signal changes across time, which could result in large flow field change if adjacent rollouts have large changes. We initialize all methods to output stochastic control signals of small noise for stable initial simulation. Our gradient-based method quickly converges to near zero after 60 epochs, while both gradient-free methods struggle in this environment. We argue that the rapid and successful convergence stems from the clear gradient provided by our method. Note that our differentiable optimization pipeline depends on the differentiability of the loss function (e.g., the imitation loss we used). This can be problematic if the objective is too complex to be characterized in a differentiable manner, in which case gradient-free methods are better alternatives. However, our framework will still excel due to its outstanding forward simulation speed, which we will elaborate next.

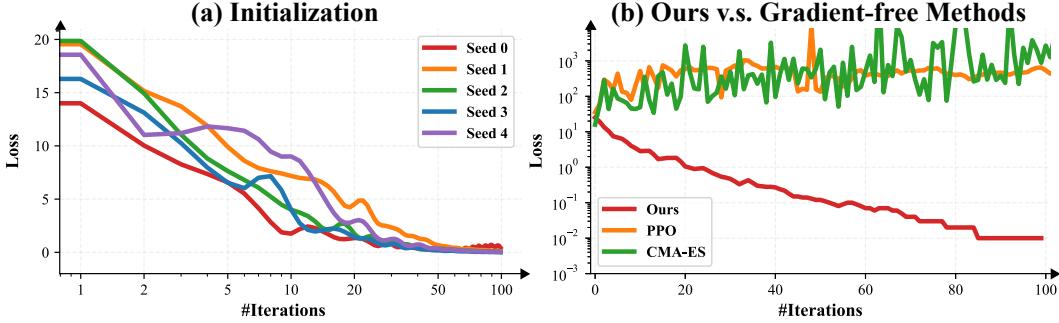


Figure 5: **Ablation Studies.** *Left:* Optimization trajectories for *Neural Heart* with 7100 parameters under different initialization. Iterations are visualized on a log scale. *Right:* Log scaled loss-iteration curves of our gradient-based method and other gradient-free optimization methods.

4.3 Time Performance Profiling and Comparison with PhiFlow

In this experiment, we demonstrate the performance efficiency of our framework through a comparison with PhiFlow. While PhiFlow operates with a TensorFlow-GPU backend, our framework is implemented in CUDA C++ and features a high-performance Geometric-Multigrid-Preconditioned-Conjugate-Gradient (MGPCG) Poisson solver [17]. To address the needs of differential operators and interpolations, which require access to neighboring cells in all directions, we divide the simulation domain into cubic blocks, each corresponding to a CUDA block. When launching a CUDA kernel, simulation data for each block is first loaded into shared memory, allowing efficient computation directly in shared memory and reducing global memory accesses. Additionally, to increase memory throughput, each block’s data is stored consecutively in global memory. Our matrix-free MGPCG solver has a faster convergence rate than PhiFlow’s Conjugate Gradient solver and uses a hierarchical grid data structure, with custom CUDA kernels for prolongation and restriction operations between coarse and fine grids.

We benchmark both the one-step forward simulation time and gradient back-propagation time at different resolutions, as shown in Table 2. The experiment runs on a workstation with an NVIDIA RTX A6000 GPU, where our framework consistently outperforms PhiFlow by an order of magnitude across all resolutions, benefiting both gradient-based and gradient-free optimization techniques. This performance improvement is particularly advantageous in fluid simulation applications, including robotics and video generation. Additionally, our system’s gym protocol compatibility [4] makes it straightforward for practitioners to integrate and test our library. We plan to release our code and documentation upon acceptance.

4.4 Memory and Time Performance Profiling and Comparison with DiffTaichi

Here we compare our solver with DiffTaichi, a differentiable programming framework, to highlight the benefits of our approach in terms of scalability and efficiency. Our method, designed specifically for differentiable fluid simulation, uses manually derived gradients, avoiding the need to store intermediate computational graph at each timestep, unlike DiffTaichi, which relies on automatic differentiation. Additionally, our adjoint derivation for the projection solve step is independent of solver iterations, making our approach well-suited for advection-projection fluid simulations. To demonstrate this, we implemented a Conjugate Gradient (CG) solver for the projection step in DiffTaichi and compared time and memory performance across four grid resolutions in a 3D optimization scenario (Table 3). Our results show that our solver requires substantially less memory, with up to 12 times less memory usage than DiffTaichi at $64 \times 64 \times 64$ resolution. This reduction in memory stems from eliminating the need to store intermediate values during each CG iteration, making our solver particularly suitable for high-resolution, long-term optimizations.

5 Related Work

Flow Control and Optimization Beginning with the pioneering work of [18], a vast literature has been devoted to the optimization of fluid systems [19]. Given a predefined design domain with

boundary conditions, a typical optimization objective is to maximize some performance functional of a fluid system (e.g., the power loss of the system) constrained by the physical equations. Similar to a conventional structural optimization problem, the design domain is discretized. The optimization algorithm decides for each element whether it should be fluid or solid to optimize some performance functions such as power loss. Examples of flow optimization applications include Stokes flow [18, 20, 21, 22], steady-state flow [23], weakly compressible flow [24], unsteady flow [25], channel flow [26], ducted flow [27], viscous flow [28], fluid-structure interaction (FSI) [29, 30, 31], fluid-thermal interaction [32, 33], microfluidics [34], aeronautics [35, 36], and aerodynamics [37, 38], to name a few. [39] developed a dynamic differentiable fluid simulator and integrated the pipeline with neural networks for learning controllers. In computer graphics, [40] developed a differentiable framework to simulate and optimize flow systems governed by design specifications with different types of boundary conditions, while [8] developed an anisotropic material model to handle different boundary conditions using topology optimization framework. Both systems focus on the Stokes flow model and have not explored applications with a dynamic flow system. [41] adapted the adjoint method to control free-surface liquids.

Differentiable Physics Simulation Differentiable simulations emerge and boost as a powerful tool to accommodate various optimization applications crossing graphics and robotics. A typical example is DiffTaichi [42], which created a differentiable programming environment to compute the gradients of physics simulations. A variety of physics simulation algorithms stemming from graphical applications have been adapted to a differentiable framework to facilitate inverse design applications, including fluids [43, 44, 45], position-based dynamics [7], cloth [46, 47], deformable objects [48], articulated bodies [49], object control [50], and solid-fluid coupling systems [39, 51]. While [51] proposed a method to differentiate Lagrangian fluid simulation, optimization of rigid geometry is not discussed. Many applications across graphics and robotics have been explored, such as soft-body design and locomotion [52, 53] and fluid manipulation [5]. However, none of these approaches focused on enabling the inverse design of fluidic device systems in dynamic Navier-Stokes flow.

Computational Design The last decade has witnessed an increasing interest in the design of computational tools and algorithms targeting the digital fabrication of physical systems. A broad range of applications have been addressed, including the mechanical characters [54, 55, 56], inflatable thin shells [57], foldable structures [58, 59], Voronoi structures [60], joints and puzzles [61], spinning objects [62], buoyancy [63], gliders [64], multicopters [65], hydraulic walkers [66], origami robots [67], articulated robots [68], and multi-material jumpers [69], to name just a few. Among these applications, the problem of optimizing the shape and control of a 3D printable object to manifest specific mechanical properties and functionalities has drawn particular attention. Examples of designing mechanical properties by optimizing materials include optics [70, 71], mechanical stability [72], strength [73], rest shape [74], and desired deformation [75, 76].

6 Conclusions, Limitation and Future Work

In this paper, we proposed a fully differentiable pipeline for neural fluidic system control and design, addressing the challenges of complex geometry representation, differentiable fluid simulation, and co-design optimization processes. Our pipeline features a low-dimensional parametric geometry representation and a differentiable Navier-Stokes simulator for predicting fluid behavior. We demonstrate the effectiveness of our pipeline in a number of complex control design tasks, ranging from different fluidic functional controls to complex neural heart control.

There are certain limitations and avenues for future work. First, the current pipeline assumes the standard Navier-Stokes model, which limits its applicability to Newtonian flow. Extending the framework to handle non-Newtonian flows or multi-physics interactions would be an interesting direction for future research. Additionally, the pipeline relies on parametric representation, which may encounter challenges in navigating complex design spaces with high-dimensional or discontinuous parameterizations such as coupling control design with topology optimization. Exploring alternative optimization algorithms or incorporating surrogate models could enhance the efficiency and robustness of the optimization process. Furthermore, while we demonstrate the effectiveness of our pipeline in several control and design tasks, additional validation and bench-marking against real-world physical experiments would be valuable to establish the pipeline's reliability and generalizability.

References

- [1] G. D. Ntouni, A. S. Lioumpas, and K. S. Nikita, “Reliable and energy-efficient communications for wireless biomedical implant systems,” *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 6, pp. 1848–1856, 2014.
- [2] D. Erickson and D. Li, “Integrated microfluidic devices,” *Analytica Chimica Acta*, vol. 507, no. 1, pp. 11–26, 2004.
- [3] K. Nonami, F. Kendoul, S. Suzuki, W. Wang, and D. Nakazawa, *Autonomous flying robots: unmanned aerial vehicles and micro aerial vehicles*. Springer Science & Business Media, 2010.
- [4] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” *arXiv preprint arXiv:1606.01540*, 2016.
- [5] Z. Xian, B. Zhu, Z. Xu, H.-Y. Tung, A. Torralba, K. Fragkiadaki, and C. Gan, “Fluidlab: A differentiable environment for benchmarking complex fluid manipulation,” *ICLR*, 2023.
- [6] P. Ma, T. Du, J. Z. Zhang, K. Wu, A. Spielberg, R. K. Katzschatmann, and W. Matusik, “Diffaqua: A differentiable computational design pipeline for soft underwater swimmers with shape interpolation,” *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, p. 132, 2021.
- [7] T. Du, K. Wu, P. Ma, S. Wah, A. Spielberg, D. Rus, and W. Matusik, “Diffpd: Differentiable projective dynamics,” *ACM Trans. Graph.*, vol. 41, no. 2, nov 2021. [Online]. Available: <https://doi.org/10.1145/3490168>
- [8] Y. Li, T. Du, S. Grama Srinivasan, K. Wu, B. Zhu, E. Sifakis, and W. Matusik, “Fluidic topology optimization with an anisotropic mixture model,” *ACM Trans. Graph.*, nov 2022. [Online]. Available: <https://doi.org/10.1145/3550454.3555429>
- [9] J. Stam, “Stable fluids,” *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, p. 121–128, 1999.
- [10] R. Bridson, *Fluid simulation for computer graphics*, 2nd ed. Boca Raton, FL, USA: AK Peters/CRC Press, 2015.
- [11] F. H. Harlow and J. E. Welch, “Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface,” *Physics of Fluids*, vol. 8, pp. 2182–2189, 1965.
- [12] Y. T. Ng, C. Min, and F. Gibou, “An efficient fluid–solid coupling algorithm for single-phase flows,” *Journal of Computational Physics*, vol. 228, no. 23, pp. 8807–8829, 2009.
- [13] W. Jakob, J. Rhinlander, and D. Moldovan, “pybind11 — seamless operability between c++11 and python,” 2016, <https://github.com/pybind/pybind11>.
- [14] 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,” in *Neural Information Processing Systems*, 2019.
- [15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017.
- [16] N. Hansen and S. Kern, “Evaluating the cma evolution strategy on multimodal test functions,” vol. 3242, 2004, pp. 282–291.
- [17] A. McAdams, E. Sifakis, and J. Teran, “A parallel multigrid poisson solver for fluids simulation on large grids,” ser. SCA ’10, 2010, p. 65–74.
- [18] T. Borrrell and J. Petersson, “Topology optimization of fluids in stokes flow,” *International journal for numerical methods in fluids*, vol. 41, no. 1, pp. 77–107, 2003.
- [19] J. Alexandersen and C. S. Andreasen, “A review of topology optimisation for fluid-based problems,” *Fluids*, vol. 5, no. 1, 2020. [Online]. Available: <https://www.mdpi.com/2311-5521/5/1/29>

- [20] J. K. Guest and J. H. Prévost, “Topology optimization of creeping fluid flows using a darcy–stokes finite element,” *International Journal for Numerical Methods in Engineering*, vol. 66, no. 3, pp. 461–484, 2006.
- [21] N. Aage, T. H. Poulsen, A. Gersborg-Hansen, and O. Sigmund, “Topology optimization of large scale stokes flow problems,” *Structural and Multidisciplinary Optimization*, vol. 35, no. 2, pp. 175–180, 2008.
- [22] V. J. Challis and J. K. Guest, “Level set topology optimization of fluids in stokes flow,” *International Journal for Numerical Methods in Engineering*, vol. 79, no. 10, pp. 1284–1308, 2009.
- [23] S. Zhou and Q. Li, “A variational level set method for the topology optimization of steady-state navier–stokes flow,” *Journal of Computational Physics*, vol. 227, no. 24, pp. 10 178–10 195, 2008.
- [24] A. Evgrafov, “Topology optimization of slightly compressible fluids,” *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik: Applied Mathematics and Mechanics*, vol. 86, no. 1, pp. 46–62, 2006.
- [25] Y. Deng, Z. Liu, and Y. wu, “Topology optimization of steady and unsteady incompressible navier–stokes flows driven by body forces,” *Structural and Multidisciplinary Optimization*, vol. 47, 11 2012.
- [26] A. Gersborg-Hansen, O. Sigmund, and R. B. Haber, “Topology optimization of channel flow problems,” *Structural and Multidisciplinary Optimization*, vol. 30, no. 3, pp. 181–192, 2005.
- [27] C. Othmer, E. de Villiers, and H. Weller, “Implementation of a continuous adjoint for topology optimization of ducted flows,” in *18th AIAA Computational Fluid Dynamics Conference*. Reston, VA, USA: the American Institute of Aeronautics and Astronautics, 2007, p. 3947.
- [28] E. Kontoleontos, E. Papoutsis-Kiachagias, A. Zymaris, D. Papadimitriou, and K. Giannakoglou, “Adjoint-based constrained topology optimization for viscous flows, including heat transfer,” *Engineering Optimization*, vol. 45, no. 8, pp. 941–961, 2013.
- [29] G. H. Yoon, “Topology optimization for stationary fluid-structure interaction problems using a new monolithic formulation,” *International Journal for Numerical Methods in Engineering*, vol. 82, no. 5, pp. 591–616, 2010.
- [30] W. J. P. Casas and R. Pavanello, “Optimization of fluid-structure systems by eigenvalues gap separation with sensitivity analysis,” *Applied Mathematical Modelling*, vol. 42, pp. 269–289, 2017.
- [31] C. S. Andreasen and O. Sigmund, “Topology optimization of fluid–structure-interaction problems in poroelasticity,” *Computer Methods in Applied Mechanics and Engineering*, vol. 258, pp. 55–62, 2013.
- [32] T. Matsumori, T. Kondoh, A. Kawamoto, and T. Nomura, “Topology optimization for fluid–thermal interaction problems under constant input power,” *Structural and Multidisciplinary Optimization*, vol. 47, no. 4, pp. 571–581, 2013.
- [33] K. Yaji, T. Yamada, S. Kubo, K. Izui, and S. Nishiwaki, “A topology optimization method for a coupled thermal–fluid problem using level set boundary expressions,” *International Journal of Heat and Mass Transfer*, vol. 81, pp. 878–888, 2015.
- [34] C. S. Andreasen, A. R. Gersborg, and O. Sigmund, “Topology optimization of microfluidic mixers,” *International Journal for Numerical Methods in Fluids*, vol. 61, no. 5, pp. 498–513, 2009.
- [35] M. Mangano, S. He, Y. Liao, D.-G. Caprace, A. Ning, and J. R. R. A. Martins, “Aeroelastic tailoring of wind turbine rotors using high-fidelity multidisciplinary design optimization,” *Wind Energy Science*, Jan. 2023, (in review).

[36] Y. Yu, Z. Lyu, Z. Xu, and J. R. R. A. Martins, “On the influence of optimization algorithm and starting design on wing aerodynamic shape optimization,” *Aerospace Science and Technology*, vol. 75, pp. 183–199, Apr. 2018.

[37] A. Jameson, “Optimum aerodynamic design using cfd and control theory,” *CFD Review*, vol. 3, 06 1995.

[38] K. Maute and M. Allen, “Conceptual design of aeroelastic structures by topology optimization,” *Structural and Multidisciplinary Optimization*, vol. 27, no. 1-2, pp. 27–42, 2004.

[39] T. Takahashi, J. Liang, Y.-L. Qiao, and M. C. Lin, “Differentiable fluids with solid coupling for learning and control,” in *AAAI*, 2021.

[40] T. Du, K. Wu, A. Spielberg, W. Matusik, B. Zhu, and E. Sifakis, “Functional optimization of fluidic devices with differentiable stokes flow,” *ACM Trans. Graph.*, vol. 39, no. 6, Dec. 2020. [Online]. Available: <https://doi.org/10.1145/3414685.3417795>

[41] A. McNamara, A. Treuille, Z. Popović, and J. Stam, “Fluid control using the adjoint method,” *ACM Trans. Graph.*, vol. 23, no. 3, p. 449–456, aug 2004. [Online]. Available: <https://doi.org/10.1145/1015706.1015744>

[42] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand, “DiffTaichi: Differentiable programming for physical simulation,” in *ICLR*, 2020.

[43] P. Holl, V. Koltun, K. Um, and N. Thuerey, “phiflow: A differentiable pde solving framework for deep learning via physical simulations,” in *Advances in Neural Information Processing Systems (NeurIPS) Workshop*, 2022.

[44] P. Holl, V. Koltun, and N. Thuerey, “Learning to control pdes with differentiable physics,” 2020.

[45] B. List, L.-W. Chen, and N. Thuerey, “Learned turbulence modelling with differentiable fluid solvers: physics-based loss functions and optimisation horizons,” *Journal of Fluid Mechanics*, vol. 949, Sep. 2022. [Online]. Available: <http://dx.doi.org/10.1017/jfm.2022.738>

[46] Y. Li, T. Du, K. Wu, J. Xu, and W. Matusik, “Diffcloth: Differentiable cloth simulation with dry frictional contact,” *ACM Trans. Graph.*, mar 2022. [Online]. Available: <https://doi.org/10.1145/3527660>

[47] Y. Li, H.-y. Chen, E. Larionov, N. Sarafianos, W. Matusik, and T. Stuyck, “DiffAvatar: Simulation-ready garment optimization with differentiable simulation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2024. [Online]. Available: <https://doi.ieeecomputersociety.org/10.1109/CVPR52733.2024.00418>

[48] Y. Qiao, J. Liang, V. Koltun, and M. Lin, “Differentiable simulation of soft multi-body systems,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[49] Y.-L. Qiao, J. Liang, V. Koltun, and M. C. Lin, “Efficient differentiable simulation of articulated bodies,” in *International Conference on Machine Learning*. PMLR, 2021, pp. 8661–8671.

[50] ——, “Scalable differentiable physics for learning and control,” in *ICML*, 2020.

[51] Z. Li, Q. Xu, X. Ye, B. Ren, and L. Liu, “Diffrr: Differentiable sph-based fluid-rigid coupling for rigid body control,” *ACM Trans. Graph.*, vol. 42, no. 6, Dec 2023. [Online]. Available: <https://doi.org/10.1145/3618318>

[52] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik, “Chainqueen: A real-time differentiable physical simulator for soft robotics,” 2018. [Online]. Available: <https://arxiv.org/abs/1810.01054>

[53] A. Spielberg, A. Zhao, Y. Hu, T. Du, W. Matusik, and D. Rus, “Learning-in-the-loop optimization: End-to-end control and co-design of soft robots through learned deep latent representations,” in *Neural Information Processing Systems*, 2019.

[54] S. Coros, B. Thomaszewski, G. Noris, S. Sueda, M. Forberg, R. W. Sumner, W. Matusik, and B. Bickel, “Computational design of mechanical characters,” *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 83, 2013.

[55] B. Thomaszewski, S. Coros, D. Gauge, V. Megaro, E. Grinspun, and M. Gross, “Computational design of linkage-based characters,” *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 64, 2014.

[56] D. Ceylan, W. Li, N. J. Mitra, M. Agrawala, and M. Pauly, “Designing and fabricating mechanical automata from mocap sequences,” *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 186, 2013.

[57] M. Skouras, B. Thomaszewski, P. Kaufmann, A. Garg, B. Bickel, E. Grinspun, and M. Gross, “Designing inflatable structures,” *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 63, 2014.

[58] S. Felton, M. Tolley, E. Demaine, D. Rus, and R. Wood, “A method for building self-folding machines,” *Science*, vol. 345, no. 6197, pp. 644–646, 2014.

[59] C. Sung and D. Rus, “Foldable joints for foldable robots,” *Journal of Mechanisms and Robotics*, vol. 7, no. 2, p. 021012, 2015.

[60] L. Lu, A. Sharf, H. Zhao, Y. Wei, Q. Fan, X. Chen, Y. Savoye, C. Tu, D. Cohen-Or, and B. Chen, “Build-to-last: strength to weight 3d printed objects,” *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 97, 2014.

[61] T. Sun, C. Zheng, Y. Zhang, C. Yin, C. Zheng, K. Zhou, Y. Yue, B. Smith, C. Batty, Z. Xu *et al.*, “Computational design of twisty joints and puzzles.” *ACM Trans. Graph.*, vol. 34, no. 4, pp. 101–1, 2015.

[62] M. Bächer, E. Whiting, B. Bickel, and O. Sorkine-Hornung, “Spin-it: optimizing moment of inertia for spinnable objects,” *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 96, 2014.

[63] L. Wang and E. Whiting, “Buoyancy optimization for computational fabrication,” *Computer Graphics Forum (Proceedings of Eurographics)*, vol. 35, no. 2, 2016.

[64] T. Martin, N. Umetani, and B. Bickel, “Omniad: data-driven omni-directional aerodynamics,” *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 113, 2015.

[65] T. Du, A. Schulz, B. Zhu, B. Bickel, and W. Matusik, “Computational multicopter design,” 2016.

[66] R. MacCurdy, R. Katzschmann, Y. Kim, and D. Rus, “Printable hydraulics: A method for fabricating robots by 3d co-printing solids and liquids,” 2016.

[67] A. Schulz, C. Sung, A. Spielberg, W. Zhao, R. Cheng, E. Grinspun, D. Rus, and W. Matusik, “Interactive robogami: An end-to-end system for design of robots with ground locomotion,” *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1131–1147, 2017.

[68] A. Spielberg, B. Araki, C. R. Sung, R. Tedrake, and D. Rus, “Functional co-optimization of articulated robots,” in *ICRA*. IEEE, 2017, pp. 5035–5042.

[69] D. Chen, D. I. Levin, W. Matusik, and D. M. Kaufman, “Dynamics-aware numerical coarsening for fabrication design,” *ACM Trans. Graph.*, vol. 34, no. 4, 2017.

[70] M. Hašan, M. Fuchs, W. Matusik, H. Pfister, and S. Rusinkiewicz, “Physical reproduction of materials with specified subsurface scattering,” *ACM Trans. Graph.*, vol. 29, no. 4, 2010.

[71] Y. Dong, J. Wang, F. Pellacini, X. Tong, and B. Guo, “Fabricating spatially-varying subsurface scattering,” *ACM Trans. Graph.*, vol. 29, no. 4, 2010.

[72] O. Stava, J. Vanek, B. Benes, N. Carr, and R. Měch, “Stress relief: improving structural strength of 3d printable objects,” *ACM Trans. Graph.*, vol. 31, no. 4, 2012.

- [73] Q. Zhou, J. Panetta, and D. Zorin, “Worst-case structural analysis,” *ACM Trans. Graph.*, vol. 32, no. 4, 2013.
- [74] X. Chen, C. Zheng, W. Xu, and K. Zhou, “An asymptotic numerical method for inverse elastic shape design,” *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2014)*, vol. 33, no. 4, Aug. 2014.
- [75] B. Bickel, M. Bächer, M. A. Otaduy, H. R. Lee, H. Pfister, M. Gross, and W. Matusik, “Design and fabrication of materials with desired deformation behavior,” *ACM Trans. Graph.*, vol. 29, no. 4, 2010.
- [76] B. Zhu, M. Skouras, D. Chen, and W. Matusik, “Two-scale topology optimization with microstructures,” *ACM Trans. Graph.*, vol. 36, no. 4, jul 2017.

A Geometry Representation Implementation

2D Geometry We define a closed 2D surface using N connected cubic Bezier curves parameterized in the polar coordinate frame. A point \mathbf{c} is defined on the surface to establish the center of a polar coordinate frame. Each Bezier curve spans an arc of $\frac{2\pi}{N}$ radians on the polar coordinate plane, and its control points are symmetrically placed, each at an angular displacement of $\frac{2\pi}{3N}$ radians from their corresponding curve endpoint. The shape of each Bezier curve, and consequently the overall surface, is manipulated via two scalar parameters, ρ_1 and ρ_2 , dictating the polar coordinate distance of the two control points. The i -th cubic Bezier curve is defined by two control points $\mathbf{p}_0 = (\rho_1^i \cos \theta, \rho_1^i \sin \theta) + \mathbf{c}$ and $\mathbf{p}_1 = (\rho_2^i \cos \theta, \rho_2^i \sin \theta) + \mathbf{c}$, where \mathbf{c} is the reference center point. The endpoints $\mathbf{e}_0, \mathbf{e}_1$ are computed by ensuring $\mathbf{e}_1^i = \mathbf{e}_0^{(i+1)\%N}$ and colinearity of each pair of $\mathbf{p}_1^i, \mathbf{e}_1^i, \mathbf{p}_0^{(i+1)\%N}$. This representation offers a compact way of defining diverse geometries.

3D Geometry We parameterize a closed 3D surface using 2D surfaces defining the key cross-sections of the geometry along the z -axis of the local object frame, where each 2D surface is parameterized by N cubic Bezier curves. The parametrization includes $z = z_0$ and z_1 , which determines the Z plane of the first and last cross-section and the parameters for each key cross-section. The i -th key cross-section is defined by the center \mathbf{c}^i and control point parameters ρ_1^i, ρ_2^i for each of $i \in [1, 2, \dots, N]$. The key cross-sections are assumed to be evenly spaced along the z -axis. Then, given $z_0 \leq z \leq z_1$, the cross-section of the closed surface at $Z = z$ is defined by interpolating the centers and control points of all key cross-sections using the interpolation scheme

$$\mathbf{c}_z = \sum_{i=1}^n \mathbf{c}^i \left(\frac{z - z_0}{z_1 - z_0} \right)^i \quad (10a)$$

$$\rho_j = \sum_{i=1}^n \rho_j^i \left(\frac{z - z_0}{z_1 - z_0} \right)^i \quad (10b)$$

B Temporal Discretization of the Governing Equation

We build the fluid simulator by leveraging the operator-splitting method [9][10]. Each simulation step comprises of advection, viscosity, and projection.

Advection We employ the semi-Lagrangian advection scheme specified in (Eqs. 11 and 12) to propagate velocity through the fluid domain:

$$\frac{\tilde{\mathbf{u}}^{n+1/2} - \mathbf{u}^n}{\Delta t/2} = -\mathbf{u}^n \cdot \nabla \mathbf{u}^n, \quad (11)$$

$$\frac{\tilde{\mathbf{u}}^{n+1} - \mathbf{u}^n}{\Delta t} = -\tilde{\mathbf{u}}^{n+1/2} \cdot \nabla \mathbf{u}^n. \quad (12)$$

Viscosity For incompressible fluid with a constant viscosity coefficient, the viscous force density is equivalent to the product of the Laplacian of velocity and the viscosity coefficient. We employ explicit time integration to update the fluid velocity in response to the viscous force.

$$\frac{\hat{\mathbf{u}}^{n+1} - \tilde{\mathbf{u}}^{n+1}}{\Delta t} = \nu \nabla^2 \tilde{\mathbf{u}}^{n+1} \quad (13)$$

Projection The projection step involves an update of the pressure and the velocity field (Eq. 14a) to ensure the satisfaction of the incompressibility condition (Eq. 14b).

$$\begin{cases} \frac{\mathbf{u}^{n+1} - \hat{\mathbf{u}}^{n+1}}{\Delta t} = -\frac{1}{\rho} \nabla p^{n+1}, \\ \nabla \cdot \mathbf{u}^{n+1} = 0. \end{cases} \quad (14a)$$

$$(14b)$$

On boundaries, the pressure is regulated by two conditions: the Dirichlet boundary condition (Eq. 15a) and the non-penetrating Neumann boundary condition (Eq. 15b) given computed geometry velocity \mathbf{u}_s^{n+1} :

$$\begin{cases} p^{n+1} = 0, & x \in \partial\Omega_f^{n+1}, \\ \mathbf{u}^{n+1} \cdot \mathbf{n} = \mathbf{u}_s^{n+1} \cdot \mathbf{n}, & x \in \partial\Omega_b^{n+1}. \end{cases} \quad (15a)$$

$$(15b)$$

The pressure field for the subsequent time step p^{n+1} is determined by solving the resultant Poisson equation (Eq. 16).

$$\frac{\Delta t}{\rho} \nabla^2 p^{n+1} = \nabla \cdot \hat{\mathbf{u}}^{n+1}. \quad (16)$$

C Additional Optimization Task Details and Visualization

C.1 Switch

We visualize the initial and optimized design for the amplifier task in Fig. 6.

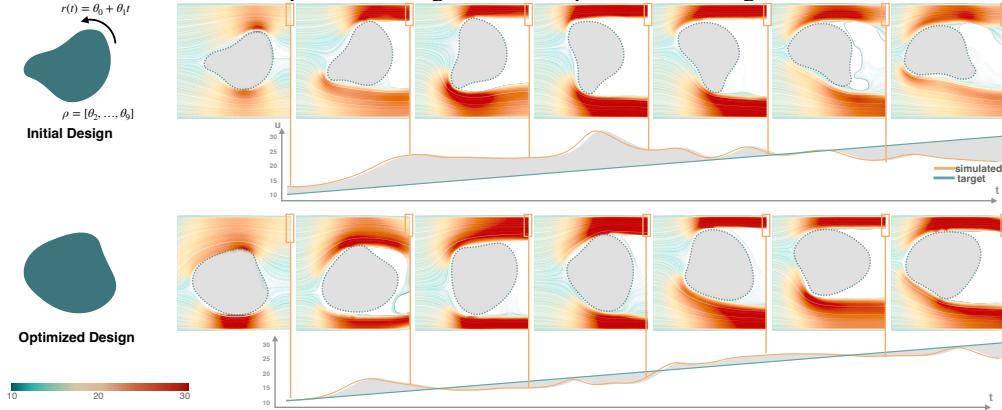


Figure 6: Fluidic Switch. The switch rotates dynamically across time (dotted lines). The shape of the switch is parameterized as a 2D Polar Bezier, whose parameters, along with the parameters of the rotation signal are subject to optimization. The top and bottom of the illustration visualize information from the initial and optimized iteration respectively. For each iteration, we visualize the design geometry (left) and corresponding streamlines of the flow field (right) at 7 key-frames evenly sampled across time. We additionally plot the target (green) and outlet velocity norm profile (orange) across time and visualize their difference in grey shaded area.

D Experiment on Fluid Solver Validation – Kármán Vortex Street

To further validate the performance of our solver, we conducted an additional experiment simulating the formation of a classic **Kármán Vortex Street**. This experiment was executed at a resolution of 512×1024 and illustrates the capability of our solver in capturing complex fluid dynamics phenomena. As shown in Fig. 7, we simulate a horizontal flow passing around a cylindrical obstacle

at three distinct kinematic viscosity values: inviscid ($\nu = 0$), moderate viscosity ($\nu = 0.01$), and high viscosity ($\nu = 0.1$).

Each viscosity setting demonstrates the characteristic vortex shedding pattern associated with Kármán Vortex Street formation. These results confirm our solver’s ability to replicate this well-known phenomenon and offer insights into the effect of varying viscosity on vortex behavior. This validation experiment supports the accuracy and versatility of the solver across different fluid conditions.

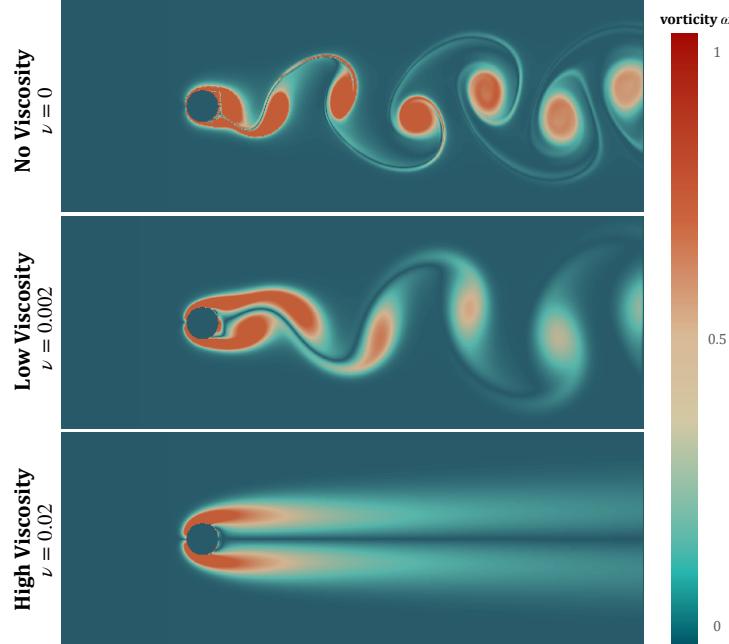


Figure 7: Solver Validation. Visualization of Karman Vortex Street under different viscosity conditions. Here, we illustrate the results of the classic Karman Vortex Street test for three different kinematic viscosity values (From top to down $\nu = 0.0$, $\nu = 0.002$ and $\nu = 0.02$), simulated using our differentiable simulator within a domain size 512×1024 . Each figure visualizes the vortex patterns, and the results demonstrate how increased viscosity leads to a notable change in vortex formation and dissipation.

E Gradient Stability and Solver Steps Statistics

Gradient stability in differentiable physics is a well-known challenge, particularly given the potential for gradient explosion or vanishing when gradients are accumulated across numerous solver steps. In our approach, however, we have not encountered significant issues with gradient stability. This stability is likely due to the accuracy of the gradients produced by our framework and the robustness of our numerical solver. To illustrate this, we provide gradient norm statistics over the full course of optimization for three tasks of varying complexity in Table 4, demonstrating consistent gradient magnitudes without evidence of explosion or vanishing. For additional robustness, our implementation includes gradient clipping with a threshold of 1.0, which can mitigate gradient explosion in particularly challenging scenarios. This technique ensures gradients remain within manageable limits and contributes to the overall stability of our optimization pipeline.

Furthermore, the actual number of solver steps required to advance between frames in our solver depends on the Courant-Friedrichs-Lowy (CFL) condition, which is maintained to ensure numerical stability. Additional statistics on solver steps over one optimization cycle across various tasks are provided in the upper portion of Table 4, offering further insight into the computational demands and stability characteristics of our approach.

Metric	Statistic	Shape Identifier	Heart 3D	Gate 3D
Steps	Mean, Std	(11, 0)	(54, 0)	(58, 0)
Gradient Norm	Min, Max Mean, Std	(0.46, 252.37) (28.04, 38.00)	(1.61, 820.82) (65.96, 109.40)	(5.13, 169.21) (40.71, 34.36)

Table 4: Statistics of the gradient norm and step count over the full course of optimization for three tasks of varying complexity.

Table 5: Gradient Validation for Shape Identifier Task

	Gradient Values by Parameter (P1 to P6)					
	P1	P2	P3	P4	P5	P6
Analytic	0.048	0.394	0.314	0.046	0.133	0.067
Finite Diff	0.048	0.396	0.314	0.046	0.134	0.066
Abs Diff	-1.7e-4	-2.1e-3	5.2e-4	-3.4e-4	-1.4e-3	7.2e-4
Elem Err	0.004	0.005	0.002	0.007	0.011	0.011
	Gradient Values by Parameter (P7 to P11)					
	P7	P8	P9	P10	P11	
Analytic	0.087	0.008	-0.022	0.540	-0.058	
Finite Diff	0.086	0.008	-0.025	0.540	-0.059	
Abs Diff	2.1e-4	-9.6e-5	2.2e-3	2.4e-4	9.5e-4	
Elem Err	0.003	0.012	0.098	0.000	0.016	

F Experiment on Gradient Validation

To ensure the correctness of the gradients in our differentiable simulation framework, we validated the analytical gradients of all kernels, functions, and the entire simulation and optimization pipeline using finite difference approximations. Specifically, we employed the central difference method with a step size of 1.2×10^{-5} to approximate the gradients numerically and compared them with the analytical gradients calculated by our solver.

In this validation experiment, we consider the end-to-end gradients for the Shape Identifier Task, which involves optimizing over 11 parameters. The analytical gradients, finite difference gradients, their absolute differences, and element-wise errors are reported in Table 5. For this task, we observed a relative error of 0.0047 for the gradient vector, confirming the high accuracy of our analytical gradients.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: The claim reflects paper's contribution and scope accurately.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: We discussed the limitations of the work in Sec. 6.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory Assumptions and Proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[NA\]](#)

Justification: We did not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental Result Reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer:[Yes]

Justification: We discussed the experimental setup in details.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We are releasing the experiment data and code on github.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental Setting/Details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide experimental setup in details.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment Statistical Significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We provide loss-iteration curves to report experiment details.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer “Yes” if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments Compute Resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: We discussed the experimental setup in the paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code Of Ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [\[Yes\]](#)

Justification: The research conform to NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader Impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [\[NA\]](#)

Justification: There is no societal impact of the work performed.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [NA]

Justification: The paper does not use existing assets.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New Assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: Our paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.