Optimal Control of PDEs Using Physics-Informed Neural Networks

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Physics-informed neural networks (PINNs) have recently become a popular method for solving forward and inverse problems governed by partial differential equations (PDEs). By incorporating the residual of the PDE into the loss function of a neural network-based surrogate model for the unknown state, PINNs can seamlessly blend measurement data with physical constraints. Here, we extend this framework to PDE-constrained optimal control problems, for which the governing PDE is fully known and the goal is to find a control variable that minimizes a desired cost objective. Importantly, we validate the performance of the PINN framework by comparing it to state-of-the-art adjoint-based optimization, which performs gradient descent on the discretized control variable while satisfying the discretized PDE. This comparison, carried out on challenging problems based on the nonlinear Kuramoto-Sivashinsky and Navier-Stokes equations, sheds light on the pros and cons of the PINN and adjoint-based approaches for solving PDE-constrained optimal control problems.

1 Introduction

In the physical sciences, data is often scarce while physical models are frequently available in the form of partial differential equations (PDEs) [11]. Leveraging these governing equations, physics-informed neural networks (PINNs) were recently proposed in [24] as a deep learning framework for solving forward and inverse problems. The basic idea behind PINNs is to approximate the solution to a given problem with a feed-forward neural network. This neural network is then trained by minimizing a composite loss function that not only penalizes the prediction error with respect to the available data but also enforces the governing equations and boundary conditions. A great benefit of PINNs is their flexibility: they can either solve forward problems in the absence of any data when the governing equations are fully known, or leverage available data to solve inverse problems involving unknown model parameters or physical quantities (for reviews on PINNs, see [17] and [2]). PINNs have since found applications in numerous fields such as fluid mechanics [25, 26, 29], heat transfer [2], solid mechanics [27, 9], medicine [28, 31], and chemistry [10].

In this paper, we investigate the potential of PINNs to solve PDE-constrained optimal control problems, for which the governing PDEs are fully known and the goal is to find a control variable that minimizes a desired cost objective. Such problems arise in a variety of fields including fluid mechanics [6], transition to turbulence [12], heat transfer [20], electromagnetism [5], topology optimization [22], and mesh refinement [15]. The control variable to optimize might represent a distributed boundary actuation, an external body force or an initial condition of the system [30, 1]. Optimal control problems are usually solved by combining gradient-descent algorithms with adjoint-based sensitivity analysis, which computes the gradient of the cost objective function with respect to the control

Submitted to the DLDE Workshop in the 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

variable using only two PDE simulations [16]. Such adjoint-based optimization frameworks are therefore very efficient when the control is a space- and/or time-dependent field, but their complexity has limited their adoption by the engineering community.

As opposed to adjoint-based optimization, a major strength of PINNs is their ease of implementation. Here, we show that the PINN framework can be readily extended to the optimal control setting by approximating the control field with its own neural network in addition to the neural network for the unknown state variable. These two networks are then simultaneously trained using a composite loss function that includes the cost objective function in addition to the PDE residual and initial/boundary conditions. A similar approach was recently proposed by [18] and [4] in the context of inverse design and parametric optimal control, respectively. In light of these recent works, the novelties of the present study are two-fold:

1. We propose a set of guidelines falling under two categories for obtaining a good optimal control solution using the PINN framework. In particular, we emphasize the importance of evaluating the cost objective using a separate forward computation with the PINN optimal control as an input.
2. We solve challenging control problems based on the Kuramoto-Sivashinsky and Navier-Stokes equations using both the PINN and adjoint-based approaches, and we discuss their respective pros and cons. This systematic comparison enables researchers to better evaluate and position PINN-based optimal control within the larger context of PDE-constrained optimization.

2 Methodology

Problem statement. In PDE-constrained optimal control, we are interested in problems of the form

$$\mathbf{c}^* = \arg\min_{\mathbf{c}} \mathcal{J}(\mathbf{u},\mathbf{c}) \quad \text{subject to } \mathcal{F}(\mathbf{u},\mathbf{c}) = 0, \tag{1}$$

where $\mathcal{J}(\mathbf{u},\mathbf{c})$ is a user-defined cost objective functional, $\mathcal{F}(\mathbf{u},\mathbf{c}) = 0$ is a PDE constraint derived from the physics of the system under consideration, $\mathbf{u}(\mathbf{x},t)$ is a space- and possibly time-dependent vector field characterizing the state of the system, and $\mathbf{c}(\mathbf{x},t)$ is a control input that might depend on space and/or time.

PINNs for optimal control of PDEs. Problem (1) can be solved using the PINN framework by introducing two neural network approximations $\mathbf{u}_{\text{NN}}(\mathbf{x},t)$ for the system state and $\mathbf{c}_{\text{NN}}(\mathbf{x},t)$ for the control input. The weights and biases $\boldsymbol{\theta}_{\mathbf{u}}$ and $\boldsymbol{\theta}_{\mathbf{c}}$ of these two neural network approximations are then found by minimizing the loss function

$$\mathcal{L}(\boldsymbol{\theta}_{\mathbf{u}},\boldsymbol{\theta}_{\mathbf{c}}) = \mathcal{L}_{\mathcal{F}}(\boldsymbol{\theta}_{\mathbf{u}},\boldsymbol{\theta}_{\mathbf{c}}) + w_{\mathcal{J}}\mathcal{L}_{\mathcal{J}}(\boldsymbol{\theta}_{\mathbf{u}},\boldsymbol{\theta}_{\mathbf{c}}), \tag{2}$$

where $\mathcal{L}_{\mathcal{F}}$ measures the mean-square error of the PDE constraint $\mathcal{F}(\mathbf{u}_{\text{NN}},\mathbf{c}_{\text{NN}}) = 0$ over a given set of residual points, $\mathcal{L}_{\mathcal{J}}$ measures the magnitude of the objective function $\mathcal{J}(\mathbf{u}_{\text{NN}},\mathbf{c}_{\text{NN}})$, and $w_{\mathcal{J}}$ is a scalar weight controlling the relative importance of $\mathcal{L}_{\mathcal{F}}$ and $w_{\mathcal{J}}$. In this way, the training process finds a control and a state that both satisfy the PDE constraint while minimizing the cost objective. However, the presence of two conflicting objectives $\mathcal{L}_{\mathcal{F}}$ and $\mathcal{L}_{\mathcal{J}}$ in the loss function (2) creates training difficulties [14]. A major contribution of the present work is therefore the following set of guidelines to help find a good optimal control solution:

1. **Validation.** We ensure that the optimal state $\mathbf{u}^*_{\text{NN}}$ found by the PINN framework reasonably satisfies the PDE and its initial/boundary conditions. This is done by first solving a forward problem based on the same PDE, which gives an idea for the magnitude of $\mathcal{L}_{\mathcal{F}}$ required for an accurate solution. Turning to the optimal control problem, a line search is performed to find the largest value of $w_{\mathcal{J}}$ that yields a similar magnitude for $\mathcal{L}_{\mathcal{F}}$ in the solution to the optimal control problem as in the forward problem.
2. **Evaluation.** We evaluate the performance of the optimal control $\mathbf{c}^*_{\text{NN}}$ found by the PINN framework by separately computing the solution of the corresponding forward problem with fixed $\mathbf{c} = \mathbf{c}^*_{\text{NN}}$. This forward solution can be performed with a PINN or a traditional solver.

Adjoint-based optimal control of PDEs. The framework of adjoint-based optimal control is a direct extension of the method of Lagrange multipliers to the case where the equality constraints are formulated as PDEs [7]. In this work, we consider as a baseline the direct-adjoint-looping (DAL) algorithm, which is a specific way to solve the optimality conditions given by the adjoint-based framework.

The PINN and adjoint-based methodologies are described in detail in Appendix A.

3 Results

Kuromoto-Sivashinsky (KS) equation. The KS equation is a nonlinear PDE serving as a prototypical model for pattern formation and chaos in physical systems. Here, we consider a problem defined over the 1D periodic domain $x \in [0, L]$ and time interval $t \in [0, T]$. Appendix B shows that starting from a given initial condition, the unforced system develops chaotic dynamics. For the optimal control problem, we thus seek a space- and time-distributed control force that drives the system state towards zero everywhere, using a cost function that penalizes the norms of both the state and the forcing over the entire spatio-temporal domain. The complete formulation of the problem can be found in Appendix B.

The solution of the problem using both the PINN and DAL approaches is described in Appendix. The optimal distributed control forces found by PINN and DAL are shown in Figures 1(a) and 1(b), respectively, and look very similar to one another. The efficacy of these control forces at driving the state towards zero is then evaluated by computing a spectral method solution of the KS equation using these control forces. The resulting state is displayed in Figures 1(c) and 1(d), showing that the PINN and DAL optimal forcings both manage to drive the state towards near-zero values. The corresponding cost objectives, calculated from the spectral solutions, have remarkably similar values of $\mathcal{J} = 20.58$ and $\mathcal{J} = 20.64$ for the PINN and DAL optimal forcings, respectively.

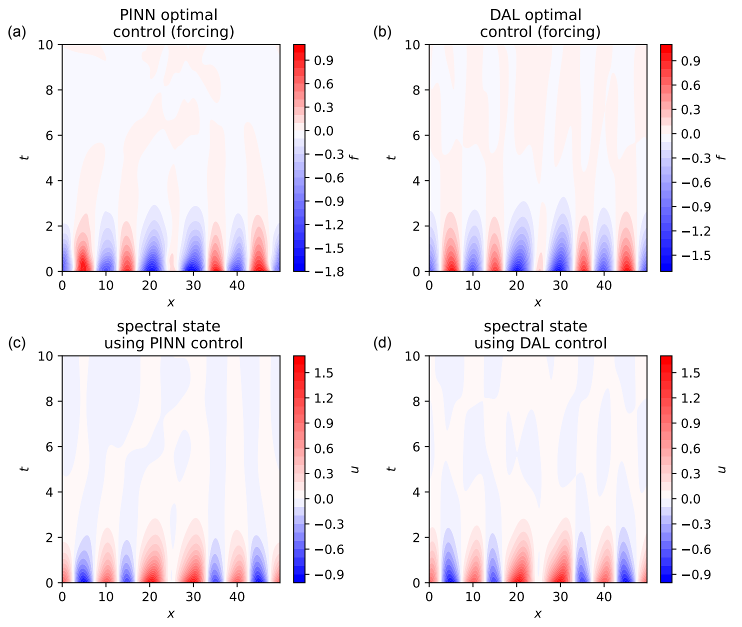

Figure 1: Optimal control of the KS equation. (a,b) Optimal forcings f^* obtained from PINN and DAL. (c,d) System states obtained from spectral method solutions of the KS equation driven by the optimal forcings f^* obtained from PINN and DAL.

Navier-Stokes (NS) equations. The NS equations are a set of coupled nonlinear PDEs describing the spatio-temporal behavior of the velocity and pressure in a fluid flow. Here, we consider a 2D horizontal channel $(x, y) \in \Omega$ with blowing and suction boundaries on the top and bottom walls. As shown in Appendix C, the flow imparted by a parabolic velocity profile at the left inlet boundary will be affected by the blowing and suction boundaries, resulting in a skewed velocity profile at the right outlet boundary. For the optimal control problem, we thus seek the velocity profile at the left inlet boundary so that the corresponding velocity profile at the outlet boundary is as close as possible to parabolic profile. The complete formulation of the problem can be found in Appendix C.

The PINN and DAL frameworks converge to rather different optimal inlet velocity profiles, shown in Figure 2(a). These profiles nonetheless share a few features: two local maxima near the centerline and around $y = 0.8$, as well as a region of negative velocity for $y < 0.3$. We evaluate the quality of these

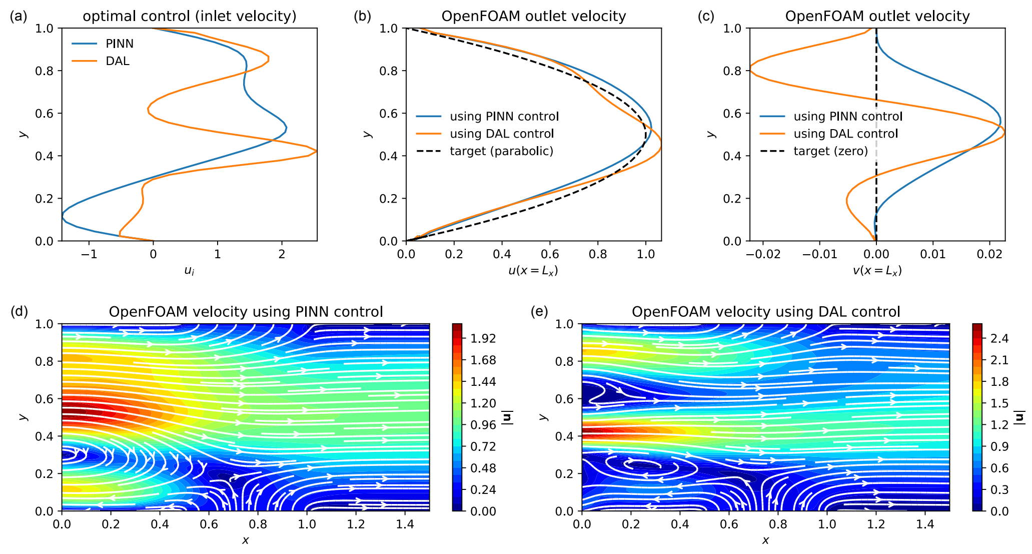

Figure 2: Optimal control of the NS equations. (a) Optimal inlet velocity profiles u_{in}^* obtained using PINN and DAL. (b,c) Outlet velocity profiles of two forward OpenFOAM solutions calculated using the optimal inlet profiles u_{in}^* from PINN and DAL, compared with the target parabolic profile. (d,e) Velocity magnitude and streamlines of the OpenFOAM solutions calculated using the optimal inlet profiles u_{in}^* from PINN and DAL.

optimal inlet profiles by computing the corresponding flow fields using the open-source finite-volume solver OpenFOAM. The resulting outlet velocity profiles, displayed in Figures 2(b,c), are both nearly parabolic with comparable cost values: $\mathcal{J} = 0.00298$ and $\mathcal{J} = 0.00265$ for the PINN and DAL inlet profiles, respectively. Yet, the PINN inlet velocity profile is smoother and produces an outlet profile that has a more parabolic shape than its DAL counterpart. Figures 2(d) and 2(e) display the velocity magnitude and streamlines of the two OpenFOAM solutions calculated using the PINN and DAL optimal inlet profiles, respectively. In both cases, the bottom region of negative inlet velocity attracts some of the fluid entering through the blowing boundary, which reduces the effect of the latter on the outlet profile.

4 Discussion and conclusions

We now draw lessons from our comparative study and assess the pros and cons of the PINN and DAL approaches. Our results demonstrate that the PINN approach can be similarly effective as DAL in solving optimal control problems. In terms of computational costs, our PINN solutions were obtained in 2 hours 4 min for the KS problem and 8 hours 20 min for the NS problem using one Tesla V100 GPU. On the other hand, the DAL solutions were obtained in 2 hours 55 min for the KS problem and 28 hours for the NS problem, using a single CPU core. Contrary to what is usually claimed in the PINN literature, DAL is therefore computationally more efficient since it achieves comparable runtimes with a single CPU core. This being said, important advantages of the PINN framework are its flexibility and ease of implementation, since it takes very little effort to adapt a PINN code for any forward problem to an optimal control problem. By contrast, the DAL approach may involve the cumbersome derivation of the adjoint equation and cost objective gradient, which needs to be repeated after any mere change of boundary conditions or cost objective function. The adjoint equation and DAL iterative procedure then need to be implemented in a numerical solver, which is no small task when complicated governing PDEs and/or complex geometries are involved. Thus, the PINN framework brings optimal control problems within reach of a much wider community compared with adjoint-based approaches such as DAL, which may compensate its lesser computational efficiency.

References

[1] Alfio Borzì and Volker Schulz. *Computational optimization of systems governed by partial differential equations*. SIAM, 2011.

[2] Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6):060801, 2021.

[3] Predrag Cvitanović, Ruslan L Davidchack, and Evangelos Siminos. On the state space geometry of the kuramoto–sivashinsky flow in a periodic domain. *SIAM Journal on Applied Dynamical Systems*, 9(1):1–33, 2010.

[4] Nicola Demo, Maria Strazzullo, and Gianluigi Rozza. An extended physics informed neural network for preliminary analysis of parametric optimal control problems. *arXiv preprint arXiv:2110.13530*, 2021.

[5] Yongbo Deng and Jan G Korvink. Self-consistent adjoint analysis for topology optimization of electromagnetic waves. *Journal of Computational Physics*, 361:353–376, 2018.

[6] Dimitry PG Foures, Colm P Caulfield, and Peter J Schmid. Optimal mixing in two-dimensional plane poiseuille flow at finite péclet number. *Journal of Fluid Mechanics*, 748:241–277, 2014.

[7] Michael B Giles and Niles A Pierce. An introduction to the adjoint approach to design. *Flow, turbulence and combustion*, 65(3):393–415, 2000.

[8] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[9] Ehsan Haghighat, Maziar Raissi, Adrian Moure, Hector Gomez, and Ruben Juanes. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379:113741, 2021.

[10] Weiqi Ji, Weilun Qiu, Zhiyu Shi, Shaowu Pan, and Sili Deng. Stiff-pinn: Physics-informed neural network for stiff chemical kinetics. *The Journal of Physical Chemistry A*, 125(36):8098–8106, 2021.

[11] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[12] RR Kerswell. Nonlinear nonmodal stability theory. *Annual Review of Fluid Mechanics*, 50:319–345, 2018.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Aditi S Krishnapriyan, Amir Gholami, Shandian Zhe, Robert M Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *arXiv preprint arXiv:2109.01050*, 2021.

[15] Shengtai Li and Linda Petzold. Adjoint sensitivity analysis for time-dependent partial differential equations with adaptive mesh refinement. *Journal of Computational Physics*, 198(1):310–325, 2004.

[16] Jacques-Louis Lions. *Optimal control of systems governed by partial differential equations*, volume 170. Springer Verlag, 1971.

[17] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[18] Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson. Physics-informed neural networks with hard constraints for inverse design. *arXiv preprint arXiv:2102.04626*, 2021.

[19] S Nabi, P Grover, and Colm-cille Patrick Caulfield. Adjoint-based optimization of displacement ventilation flow. *Building and Environment*, 124:342–356, 2017.

[20] Saleh Nabi, Piyush Grover, and CP Caulfield. Nonlinear optimal control strategies for buoyancy-driven flows in the built environment. *Computers & Fluids*, 194:104313, 2019.

[21] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[22] E Oktay, HU Akay, and O Merttopcuoglu. Parallelized structural topology optimization and cfd coupling for design of aircraft wing structures. *Computers & Fluids*, 49(1):141–145, 2011.

[23] Suhas V Patankar and D Brian Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International journal of heat and mass transfer*, 15(10):1787–1806, 1972.

[24] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[25] Maziar Raissi, Zhicheng Wang, Michael S Triantafyllou, and George Em Karniadakis. Deep learning of vortex-induced vibrations. *Journal of Fluid Mechanics*, 861:119–137, 2019.

[26] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

[27] Chengping Rao, Hao Sun, and Yang Liu. Physics-informed deep learning for computational elastodynamics without labeled data. *Journal of Engineering Mechanics*, 147(8):04021043, 2021.

[28] Francisco Sahli Costabal, Yibo Yang, Paris Perdikaris, Daniel E Hurtado, and Ellen Kuhl. Physics-informed neural networks for cardiac activation mapping. *Frontiers in Physics*, 8:42, 2020.

[29] Luning Sun, Han Gao, Shaowu Pan, and Jian-Xun Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020.

[30] Fredi Tröltzsch. *Optimal control of partial differential equations: theory, methods, and applications*, volume 112. American Mathematical Soc., 2010.

[31] Rudolf LM van Herten, Amedeo Chiribiri, Marcel Breeuwer, Mitko Veta, and Cian M Scannell. Physics-informed neural networks for myocardial perfusion mri quantification. *arXiv preprint arXiv:2011.12844*, 2020.

A Methodology details

In order to provide additional details on the PINN and adjoint-based methodologies for solving the control problem (1), we need to specify the equations contained in the PDE constraint $\mathcal{F}(\mathbf{u}, \mathbf{c}) = 0$. These are

$$\mathcal{R}[\mathbf{u}(\mathbf{x}, t); \mathbf{c}_v(\mathbf{x}, t)] = 0, \quad \mathbf{x} \in \Omega, t \in [0, T], \tag{3a}$$

$$\mathcal{B}[\mathbf{u}(\mathbf{x}, t); \mathbf{c}_b(\mathbf{x}, t)] = 0, \quad \mathbf{x} \in \partial\Omega, t \in [0, T], \tag{3b}$$

$$\mathcal{I}[\mathbf{u}(\mathbf{x}, 0); \mathbf{c}_0(\mathbf{x})] = 0, \quad \mathbf{x} \in \Omega, \tag{3c}$$

where $\mathcal{R}$ is the residual of the PDE, $\mathcal{B}$ are the boundary conditions, $\mathcal{I}$ is the initial condition, $\Omega \subset \mathbb{R}^d$ is the domain over which the problem is defined, and $[0, T]$ is the time window of interest. The control input $\mathbf{c}$ consists of $\mathbf{c}_v$, $\mathbf{c}_b$ and $\mathbf{c}_0$, which are respectively volume, boundary and initial control.

A.1 PINN for optimal control of PDEs.

As described in the main text, the PINN approach is based on the construction of two neural network approximations $\mathbf{u}_{\text{NN}}(\mathbf{x}, t)$ for the system state and $\mathbf{c}_{\text{NN}}(\mathbf{x}, t)$ for the control input. The loss function (2) is then used to train the weights and biases $\boldsymbol{\theta}_\mathbf{u}$ and $\boldsymbol{\theta}_\mathbf{c}$ of these two neural networks. Assuming for clarity of exposure that we only have volume control, that is, $\mathbf{c} = \mathbf{c}_v$, the term $\mathcal{L}_\mathcal{F}$ in (2) is given by

$$\mathcal{L}_\mathcal{F}(\boldsymbol{\theta}_\mathbf{u}, \boldsymbol{\theta}_\mathbf{c}) = \frac{w_r}{N_r} \sum_{i=1}^{N_r} |\mathcal{F}[\mathbf{u}_{\text{NN}}(\mathbf{x}_i^r, t_i^r; \boldsymbol{\theta}_\mathbf{u}); \mathbf{c}_{\text{NN}}(\mathbf{x}_i^r, t_i^r; \boldsymbol{\theta}_\mathbf{c})]|^2 + \frac{w_b}{N_b} \sum_{i=1}^{N_b} |\mathcal{B}[\mathbf{u}_{\text{NN}}(\mathbf{x}_i^b, t_i^b; \boldsymbol{\theta}_\mathbf{u})]|^2 + \frac{w_0}{N_0} \sum_{i=1}^{N_0} |\mathcal{I}[\mathbf{u}_{\text{NN}}(\mathbf{x}_i^0, 0; \boldsymbol{\theta}_\mathbf{u})]|^2, \boldsymbol{\theta}_\mathbf{c}), \tag{4}$$

where $\{\mathbf{x}_i^r, t_i^r\}_{i=1}^{N_r} \in \Omega$, $\{\mathbf{x}_i^b, t_i^b\}_{i=1}^{N_b} \in \partial\Omega$, $\{\mathbf{x}_i^0\}_{i=1}^{N_0} \in \Omega$ are sets of residual points over which to enforce the PDE residual (3a), boundary conditions (3b), and initial condition (3c), respectively, and w_r, w_b, w_0 are scalar weights for the different terms. The term $\mathcal{L}_\mathcal{J}(\boldsymbol{\theta}_\mathbf{u}, \boldsymbol{\theta}_\mathbf{c})$ consists of a Monte-Carlo approximation of the integral usually present in $\mathcal{J}(\mathbf{u}, \mathbf{c})$, using the same sets of residual points. We normalize the input $(\mathbf{x}, t)$ before passing it to the first layer of the neural networks $\mathbf{u}_{\text{NN}}$ and $\mathbf{c}_{\text{NN}}$.

We select the $\tanh$ activation function for $\mathbf{u}_{\text{NN}}$ and $\mathbf{c}_{\text{NN}}$, use Glorot initialization of the parameters [8], and employ the Adam optimizer [13] to find optimum values $(\boldsymbol{\theta}_\mathbf{u}^*, \boldsymbol{\theta}_\mathbf{c}^*)$ that minimize (4). At each iteration k, the parameters from both networks are concurrently updated as

$$\boldsymbol{\theta}_\mathbf{u}^{k+1} = \boldsymbol{\theta}_\mathbf{u}^k - \alpha(k)\nabla_{\boldsymbol{\theta}_\mathbf{u}} \mathcal{L}(\boldsymbol{\theta}_\mathbf{u}^k, \boldsymbol{\theta}_\mathbf{c}^k), \tag{5a}$$

$$\boldsymbol{\theta}_\mathbf{c}^{k+1} = \boldsymbol{\theta}_\mathbf{c}^k - \alpha(k)\nabla_{\boldsymbol{\theta}_\mathbf{c}} \mathcal{L}(\boldsymbol{\theta}_\mathbf{u}^k, \boldsymbol{\theta}_\mathbf{c}^k), \tag{5b}$$

where $\alpha(k)$ is an adaptive learning rate set by the Adam optimizer. At the end of the training procedure, the trained neural networks $\mathbf{u}_{\text{NN}}(\mathbf{x}, t; \boldsymbol{\theta}_\mathbf{u}^*)$ and $\mathbf{c}_{\text{NN}}(\mathbf{x}, t; \boldsymbol{\theta}_\mathbf{c}^*)$ approximately solve the optimal control problem (1).

A.2 Adjoint-based optimal control

The framework of adjoint-based optimal control is a direct extension of the method of Lagrange multipliers for constrained optimization to the case where the equality constraints are formulated as PDEs [16]. Applying this method to problem (1), one first constructs the Lagrangian

$$\mathcal{L}(\mathbf{u}, \mathbf{c}, \boldsymbol{\lambda}) = \mathcal{J}(\mathbf{u}, \mathbf{c}) - \langle \boldsymbol{\lambda}, \mathcal{F}[\mathbf{u}; \mathbf{c}] \rangle, \tag{6}$$

where $\mathbf{u}$ is required to satisfy the boundary and initial conditions (3b) and (3c), $\boldsymbol{\lambda} = \boldsymbol{\lambda}(\mathbf{x}, t)$ is the Lagrange multiplier or adjoint field, and the inner product $\langle \cdot, \cdot \rangle$ is defined as

$$\langle \mathbf{a}, \mathbf{b} \rangle = \int_0^T \int_\Omega \mathbf{a}(\mathbf{x}, t)^\mathsf{T} \mathbf{b}(\mathbf{x}, t) d\mathbf{x} dt. \tag{7}$$

Then, problem (1) is equivalent to the unconstrained problem

$$\mathbf{u}^*, \mathbf{c}^*, \boldsymbol{\lambda}^* = \arg \min_{\mathbf{u}, \mathbf{c}, \boldsymbol{\lambda}} \mathcal{L}(\mathbf{u}, \mathbf{c}, \boldsymbol{\lambda}), \tag{8}$$

whose solution is given by the stationary point(s) of the Lagrangian. This yields the relations

$$\left\langle \frac{\partial \mathcal{L}}{\partial \mathbf{u}}, \delta \mathbf{u} \right\rangle = 0 \quad \forall\, \delta \mathbf{u}, \tag{9a}$$

$$\left\langle \frac{\partial \mathcal{L}}{\partial \mathbf{c}}, \delta \mathbf{c} \right\rangle = 0 \quad \forall\, \delta \mathbf{c}, \tag{9b}$$

$$\left\langle \frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}}, \delta \boldsymbol{\lambda} \right\rangle = 0 \quad \forall\, \delta \boldsymbol{\lambda}, \tag{9c}$$

where the admissible variation $\mathbf{u} + \delta\mathbf{u}$ has to satisfy the boundary and initial conditions (3b) and (3c). The Fréchet derivative $\langle \partial\mathcal{L}/\partial\mathbf{u}, \cdot\rangle$ is defined so that

$$\left\langle \frac{\partial \mathcal{L}}{\partial \mathbf{u}}, \delta \mathbf{u} \right\rangle = \lim_{\epsilon \to 0} \frac{\mathcal{L}(\mathbf{u} + \epsilon\delta\mathbf{u}, \mathbf{c}, \boldsymbol{\lambda}) - \mathcal{L}(\mathbf{u}, \mathbf{c}, \boldsymbol{\lambda})}{\epsilon} \quad \forall\, \delta\mathbf{u}, \tag{10}$$

and similarly for $\langle \partial\mathcal{L}/\partial\mathbf{c}, \cdot\rangle$ and $\langle \partial\mathcal{L}/\partial\boldsymbol{\lambda}, \cdot\rangle$. Expanding the stationarity conditions (9) leads to

$$\left\langle \frac{\partial \mathcal{L}}{\partial \mathbf{u}}, \delta \mathbf{u} \right\rangle = \left\langle \frac{\partial \mathcal{J}}{\partial \mathbf{u}}, \delta \mathbf{u} \right\rangle - \left\langle \boldsymbol{\lambda}, \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta \mathbf{u} \right\rangle = \left\langle \frac{\partial \mathcal{J}}{\partial \mathbf{u}} - \frac{\partial \mathcal{F}}{\partial \mathbf{u}}^{\dagger} \boldsymbol{\lambda}, \delta \mathbf{u} \right\rangle = 0 \quad \forall\, \delta \mathbf{u}, \tag{11a}$$

$$\left\langle \frac{\partial \mathcal{L}}{\partial \mathbf{c}}, \delta \mathbf{c} \right\rangle = \left\langle \frac{\partial \mathcal{J}}{\partial \mathbf{c}}, \delta \mathbf{c} \right\rangle - \left\langle \boldsymbol{\lambda}, \frac{\partial \mathcal{F}}{\partial \mathbf{c}} \delta \mathbf{c} \right\rangle = \left\langle \frac{\partial \mathcal{J}}{\partial \mathbf{c}} - \frac{\partial \mathcal{F}}{\partial \mathbf{c}}^{\dagger} \boldsymbol{\lambda}, \delta \mathbf{c} \right\rangle = 0 \quad \forall\, \delta \mathbf{c}, \tag{11b}$$

$$\left\langle \frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}}, \delta \boldsymbol{\lambda} \right\rangle = - \langle \delta\boldsymbol{\lambda}, \mathcal{F} \rangle = 0 \quad \forall\, \delta \boldsymbol{\lambda}, \tag{11c}$$

where we have defined the adjoint $\mathcal{A}^{\dagger}$ of a linear operator $\mathcal{A}$ as

$$\langle \mathbf{a}, \mathcal{A}\mathbf{b} \rangle = \langle \mathcal{A}^{\dagger}\mathbf{a}, \mathbf{b} \rangle \quad \forall\, \mathbf{a}, \mathbf{b}, \tag{12}$$

where $\mathbf{a}$ satisfies the boundary conditions carried by the operator $\mathcal{A}$. The process of finding the adjoint operator $\mathcal{A}^{\dagger}$ involves integration by part and yields terminal and boundary conditions for the adjoint field $\mathbf{b}$. Thus, satisfying (11a) for given $\mathbf{u}$ and $\mathbf{c}$ gives the adjoint equation

$$\frac{\partial \mathcal{J}(\mathbf{u}, \mathbf{c})}{\partial \mathbf{u}} - \frac{\partial \mathcal{F}[\mathbf{u}, \mathbf{c}]}{\partial \mathbf{u}}^{\dagger} \boldsymbol{\lambda} = 0, \tag{13}$$

for the adjoint field $\boldsymbol{\lambda}$, with associated terminal and boundary conditions. The third stationary condition (11c) simply enforces the governing equation (3) for $\mathbf{u}$ given $\mathbf{c}$, that is,

$$\mathcal{F}[\mathbf{u}, \mathbf{c}] = 0, \tag{14}$$

with associated initial and boundary conditions. When (11a) and (11c) are satisfied, we have $\mathcal{J} = \mathcal{L}$, and (11b) therefore gives the total gradient of the cost objective with respect to the control $\mathbf{c}$,

$$\frac{\mathrm{d}\mathcal{J}(\mathbf{u}, \mathbf{c})}{\mathrm{d}\mathbf{c}} = \frac{\partial \mathcal{L}(\mathbf{u}, \mathbf{c})}{\partial \mathbf{c}} = \frac{\partial \mathcal{J}(\mathbf{u}, \mathbf{c})}{\partial \mathbf{c}} - \frac{\partial \mathcal{F}[\mathbf{u}, \mathbf{c}]}{\partial \mathbf{c}}^{\dagger} \boldsymbol{\lambda}. \tag{15}$$

For the optimal solution, $\mathrm{d}\mathcal{J}(\mathbf{u}^*, \mathbf{c}^*)/\mathrm{d}\mathbf{c} = 0$ holds.

There exists various adjoint-based algorithms for obtaining the optimal solution $\mathbf{u}^*, \mathbf{c}^*, \boldsymbol{\lambda}^*$ given by the stationarity conditions (11). These algorithms solve the same set of equations, namely the direct (forward) PDE (14) and adjoint PDE (13), to determine the sensitivity of the cost function to the design parameters, given by (15). The difference is, however, in the manner by which the optimal solution is obtained by each algorithm. In this work, we use the direct-adjoint-looping (DAL) algorithm [6, 19, 20], which proceeds as follows. At each iteration k, one first solves the forward PDE (14) for $\mathbf{u}^k$, given the current control $\mathbf{c}^k$. With $\mathbf{u}^k$ and $\mathbf{c}^k$ in hand, one then solves the adjoint PDE (13) for $\boldsymbol{\lambda}^k$ in backward time since the adjoint PDE contains a terminal condition instead of an initial condition. Finally, one computes the gradient of the cost objective using (15), which is then used to update the control as

$$\mathbf{c}^{k+1} = \mathbf{c}^k - \beta \frac{\mathrm{d}\mathcal{J}(\mathbf{u}^k, \mathbf{c}^k)}{\mathrm{d}\mathbf{c}}, \tag{16}$$

with β a learning rate that we will keep fixed. It should be noted that the convergence rate can be increased by employing more sophisticated update formulas such as quasi-Newton methods [21]. In our case, every gradient update only requires two PDE solutions, one for the forward PDE and one for the adjoint PDE. We end the iterations once the cost objective has stopped decreasing, or alternatively once the gradient (15) becomes small enough.

B Kuramoto-Sivashinski equation

B.1 Problem formulation

The Kuramoto-Sivashinsky (KS) equation is one of the simplest PDEs that generates chaotic behavior. It takes the form

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{\partial^2 u}{\partial x^2} + \frac{\partial^4 u}{\partial x^4} = f(x,t), \tag{17}$$

where $u(x,t)$ is the velocity at position $x \in [0,L]$ and time $t \in [0,T]$, and $f(x,t)$ is a distributed control force. We use periodic boundary conditions and choose as initial condition

$$u_0(x) = \cos\left(\frac{2\pi x}{10}\right) + \operatorname{sech}\left(\frac{x-L/2}{5}\right). \tag{18}$$

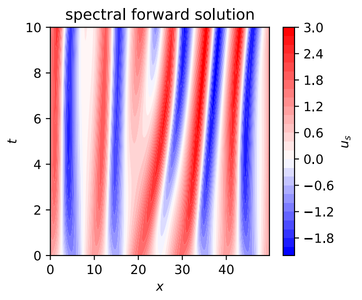

Figure 3: Spectral method simulation of the forward KS equation without forcing.

Solutions to the KS equation without forcing undergo a sequence of bifurcations as L increases; the zero state is a stable solution for $L < 2\pi$ but becomes linearly unstable for $L > 2\pi$, while for even larger L the solution becomes chaotic. Here, we choose $L = 50$ which corresponds to the chaotic regime [3] as shown in Figure 3.

For the optimal control problem, we thus seek the control force $f(x,t)$ that drives the state towards zero everywhere. We formulate the optimal control problem as

$$f^* = \arg\min_f \mathcal{J}(u,f) \quad \text{subject to (17)}, \tag{19}$$

where the objective cost is

$$\mathcal{J}(u,f) = \frac{1}{2}\int_0^T \int_0^L (|u(x,t)|^2 + \sigma|f(x,t)|^2)dxdt. \tag{20}$$

Thus, we seek the optimal forcing $f^*(x,t)$ that drives the unstable state to zero by minimizing a quadratic cost that balances the norms of both the state and the forcing. This quadratic cost functional is widespread in control theory; here we choose $\sigma = 1$.

B.2 Implementation of the PINN solution

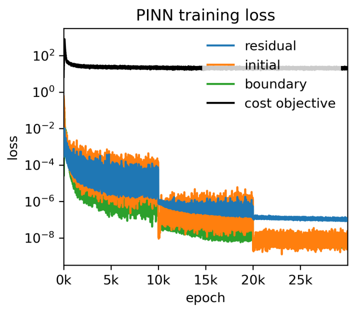

Figure 4: Loss components during training of the PINN solution.

To solve this problem in the PINN framework, we represent $u(x,t)$ and $f(x,t)$ with two neural networks consisting of 5 hidden layers of 50 neurons each. We sample $N_r = 80000$ residual training points $(x_i,t_i) \in [0,L] \times [0,T]$ using Latin Hypercube Sampling (LHS). We select $N_b = 82$ equally-spaced boundary training points $(x_i,t_i) \in \{0,L\} \times [0,T]$, and $N_0 = 41$ equally-spaced initial training points $(x_i,t_i) \in [0,L] \times \{0\}$. We then train both networks simultaneously using the loss (2). At the beginning of each epoch, the entire set of N_r residual points is shuffled and divided into 20 minibatches of $N_r/20 = 4000$ points each. We evaluate the integral in the cost objective (20) using Monte Carlo integration with the same minibatch of residual training points used in evaluating the residual loss component. We use the scalar weights $w_r = w_b = w_0 = 1$ and $w_{\mathcal{J}} = 10^{-3}$, choose an initial learning rate of 10^{-3} and decrease it by a factor 10 after 10k and 20k epochs of training, for a total of 30k epochs. The convergence of the loss components during training are shown in Figure 4.

B.3 Implementation of the adjoint-based solution

The adjoint KS equation and the gradient of the cost objective can be derived by applying the methodology outlined in Appendix A.2. The adjoint KS equation, obtained from (13) using integration by part, is

$$-\frac{\partial \lambda}{\partial t} - u\frac{\partial \lambda}{\partial x} + \frac{\partial^2 \lambda}{\partial x^2} + \frac{\partial^4 \lambda}{\partial x^4} = -u(x,t), \tag{21}$$

where $\lambda(x,t)$ is the adjoint field and $u(x,t)$ is the forward field that solves the KS equation (17) given the control (forcing) $f(x,t)$. The adjoint equation is supplemented with periodic boundary conditions and the terminal condition $\lambda(x,T) = 0$. Finally, the total gradient of the cost objective with respect to the control $f(x,t)$, obtained from (15), is

$$\frac{\mathrm{d}\mathcal{J}(u,f)}{\mathrm{d}f} = 2\sigma f(x,t) - \lambda(x,t). \tag{22}$$

The DAL optimal solution is obtained by iteratively solving the KS equation and its adjoint, updating the control $f(x,t)$ at each iteration with the gradient descent formula (16). A spectral solver with 256 Fourier modes and semi-implicit Euler scheme with $dt = 10^{-4}$ is used to solve the forward and adjoint KS equations. We start the iterations with a zero initial guess for the control $f(x,t)$ and employ a learning rate $\beta = 0.001$. The convergence of the cost objective during the DAL iterations is shown in Figure 5.

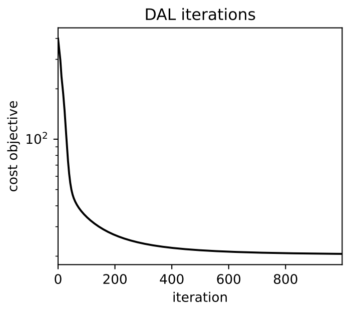

Figure 5: Cost objective during DAL iterations.

C Navier-Stokes equations

C.1 Problem formulation

We consider the steady 2D incompressible Navier-Stokes (NS) equations in the geometry depicted in Figure 6. In non-dimensional form, these equations are expressed as

$$(\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla p + \frac{1}{Re}\nabla^2\mathbf{u}, \tag{23a}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{23b}$$

where the velocity field $\mathbf{u}(\mathbf{x}) = (u(x,y), v(x,y))$ and pressure field $p(\mathbf{x}) = p(x,y)$ are defined in the rectangular 2D domain $\Omega = (L_x, L_y) = (1.5, 1)$, and we choose the Reynolds number $Re = 100$. The boundary conditions for the velocity are

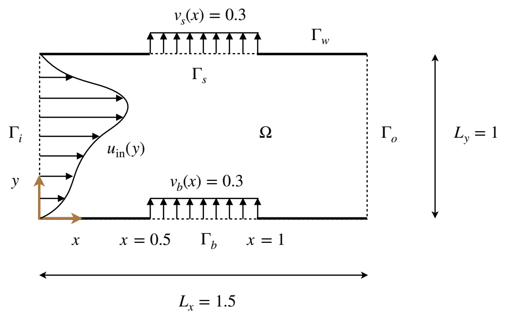

Figure 6: Setup of the domain.

$$\mathbf{u} = (u_{\text{in}}(y), 0) \quad \text{on } \Gamma_i, \tag{24a}$$

$$\mathbf{u} = (v_b(x), 0) \quad \text{on } \Gamma_b, \tag{24b}$$

$$\mathbf{u} = (v_s(x), 0) \quad \text{on } \Gamma_s, \tag{24c}$$

$$(\mathbf{n}\cdot\nabla)\mathbf{u} = (0, 0) \quad \text{on } \Gamma_o, \tag{24d}$$

$$\mathbf{u} = (0, 0) \quad \text{on } \Gamma_w, \tag{24e}$$

while the boundary conditions for the pressure are

$$(\mathbf{n}\cdot\nabla)p = 0 \quad \text{on } \Gamma_i \cup \Gamma_b \cup \Gamma_s \cup \Gamma_w, \tag{25a}$$

$$p = p_a \quad \text{on } \Gamma_o, \tag{25b}$$

where $\mathbf{n}$ denotes the unit surface normal, and p_a is a reference pressure that we set to zero. These boundary conditions correspond to a prescribed horizontal velocity profile $u_{\text{in}}(y)$ at an inlet Γ_i, a prescribed velocity profile $v_b(x)$ and $v_s(x)$ at two blowing and suction boundaries Γ_b and Γ_s, an outflow boundary Γ_o and no-slip walls Γ_w. We choose $v_b(x) = v_s(x) = 0.3$, and $u_{\text{in}}(y)$ will be specified later. Although pressure boundary conditions are not usually stated explicitly, they do affect the solution and we will therefore implement the same boundary conditions for the PINN and the adjoint-based solutions.

Using a parabolic inlet velocity profile $u_{\text{in}}(y) = u_{\text{parab}}(y) = 4y(1-y)/L_y^2$ produces the flow field shown in Figure 7, obtained with the finite-volume code OpenFOAM on a mesh of size 400 elements (20×20). The finite-volume solution is carried out using the icoFOAM solver, which implements the SIMPLE algorithm [23] for pressure and velocity decoupling. We observe that the blowing and suction boundaries deflect the parabolic inlet profile as the fluid moves through the channel, resulting in a skewed velocity profile at the outlet, reaching its maximum value at $y = 0.7$.

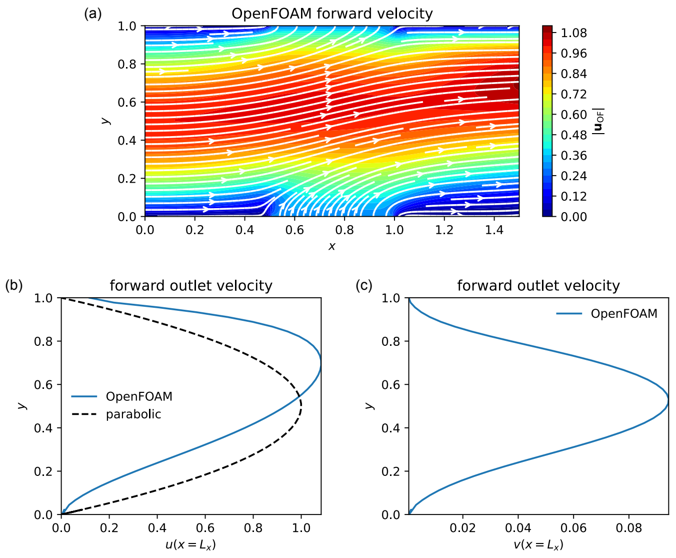

Figure 7: OpenFOAM forward solution of the Navier-Stokes equations. (a) Velocity magnitude and streamlines. (b,c) Velocity profile at the outlet Γ_o.

Is it then possible to find an inlet velocity profile $u_{\text{in}}(y)$ so that the outlet velocity profile is close to parabolic? This motivates the control problem

$$u_{\text{in}}^* = \arg\min_{u_{\text{in}}} \mathcal{J}(\mathbf{u}) \quad \text{subject to (23), (24) and (25),} \tag{26}$$

where the objective cost is

$$\mathcal{J}(\mathbf{u}) = \frac{1}{2}\int_0^{L_y} (|u(L_x, y) - u_{\text{parab}}(y)|^2 + |v(L_x, y)|^2)dy, \quad u_{\text{parab}}(y) = \frac{4}{L_y^2} y(1-y). \tag{27}$$

C.2 Implementation of the PINN solution

The PINN solution is obtained by representing $u(x, y)$, $v(x, y)$, $p(x, y)$ with a single network containing 5 hidden layers of 50 neurons each, and the control inlet profile $u_i(x, t)$ with a another neural network consisting of 3 hidden layers of 30 neurons each. We sample $N_r = 40000$ residual training points $(x_i, y_i) \in \Omega$ using LHS with 30000 points distributed in the entire domain and 10000 points distributed in 4 boxes of size 0.1×0.02 adjacent to the endpoints of Γ_b and Γ_s. We select $N_b = 328$ boundary training points (x_i, y_i), 82 of them equally spaced along the vertical boundaries Γ_i and Γ_o, and the rest equally spaced along the horizontal boundaries Γ_b, Γ_s, and Γ_w. We then train both networks simultaneously using the loss (2). At the beginning of each epoch, the entire set of N_r residual points is shuffled and divided into 10 minibatches of $N_r/10 = 4000$ points each. We evaluate the integral in the cost objective (20) using the midpoint rule at $N_{\mathcal{J}} = 41$ equally-spaced training points on the outflow boundary Γ_o. We use the scalar weights $\lambda_r^{u\text{-mom}} = 1$, $\lambda_r^{v\text{-mom}} = 2$, $\lambda_r^{\text{cont}} = 1$, $w_b = 100$, and $w_{\mathcal{J}} = 3$. We choose an initial learning rate of 10^{-3} and decrease it by a factor 10 after 100k and 200k epochs of training, for a total of 300k epochs. The convergence of the loss components during training are shown in Figure 8.

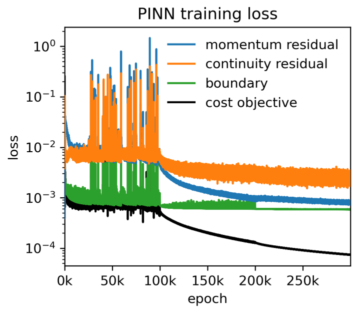

Figure 8: Loss components during training of the PINN solution.

C.3 Implementation of the adjoint-based solution

The adjoint NS equation and the gradient of the cost objective can be derived by applying the methodology outlined in Appendix A.2. We use the Einstein notation so that the velocity field will be denoted $\mathbf{u}(\mathbf{x}) = (u_1(x_1,x_2), u_2(x_1,x_2))$. The augmented objective functional, i.e. Lagrangian, corresponding to the control problem (26) is

$$\mathcal{L} = \mathcal{J} + \left\langle \lambda_i, \frac{\partial u_i u_j}{\partial x_j} + \frac{\partial p_i}{\partial x_i} - \frac{1}{Re}\frac{\partial^2 u_i}{\partial x_j^2} \right\rangle + \left\langle \Pi, -\frac{\partial u_j}{\partial x_j} \right\rangle, \tag{28}$$

where $\boldsymbol{\lambda} = (\lambda_1(x_1,x_2), \lambda_2(x_1,x_2))$ and Π are adjoint velocity and pressure fields, respectively, and the inner product $\langle \cdot, \cdot \rangle$ is defined as

$$\langle a, b \rangle = \int_\Omega a(\mathbf{x}) b(\mathbf{x}) dV. \tag{29}$$

The variation of the Lagrangian is

$$\delta\mathcal{L} = \delta\mathcal{J} + \left\langle \lambda_i, \delta u_j \frac{\partial u_i}{\partial x_j} + u_j \frac{\partial \delta u_i}{\partial x_j} + \frac{\partial \delta p_i}{\partial x_i} - \frac{1}{Re}\frac{\partial^2 \delta u_j}{\partial x_j^2} \right\rangle + \left\langle \Pi, -\frac{\partial \delta u_j}{\partial x_j} \right\rangle. \tag{30}$$

For optimality $\delta\mathcal{L} = 0$ should be satisfied. Using vector calculus and integration by parts for each term, appropriate Euler-Lagrange equations can be derived. For instance,

$$\left\langle \lambda_i, \delta u_j \frac{\partial u_i}{\partial x_j} \right\rangle = \left\langle \delta u_i, \lambda_j \frac{\partial u_j}{\partial x_i} \right\rangle \tag{31a}$$

$$\left\langle \lambda_i, u_j \frac{\partial \delta u_i}{\partial x_j} \right\rangle = -\left\langle \delta u_i, u_j \frac{\partial \lambda_i}{\partial x_j} \right\rangle + \int_{\partial\Omega} \lambda_i \delta u_i u_j n_j dS \tag{31b}$$

$$\left\langle \lambda_i, \frac{1}{Re}\frac{\partial^2 \delta u_i}{\partial x_j^2} \right\rangle = \left\langle \delta u_i, \frac{1}{Re}\frac{\partial^2 \lambda_i}{\partial x_j^2} \right\rangle + \int_{\partial\Omega} \frac{1}{Re}\left(n_j \frac{\partial \lambda_i}{\partial x_j}\delta u_i - n_j \frac{\partial \delta u_i}{\partial x_j}\lambda_i \right) dS, \tag{31c}$$

and so on for the other terms. Here, $\mathbf{n} = (n_1, n_2)$ is the normal unit vector of the surface. From the volumetric integrals, the adjoint equations are recovered as

$$\lambda_j \frac{\partial u_j}{\partial x_i} - u_j \frac{\partial \lambda_i}{\partial x_j} - \frac{1}{Re}\frac{\partial^2 \lambda_i}{\partial x_j^2} + \frac{\partial \Pi}{\partial x_i} = 0, \tag{32a}$$

$$\frac{\partial \lambda_j}{\partial x_j} = 0. \tag{32b}$$

Setting surface integrals to zero and decomposing these integrals into normal and tangential components, we obtain the corresponding boundary conditions for the adjoint velocity and pressure as

$$\lambda_1 = \lambda_2 = 0, (\mathbf{n}\cdot\nabla)\Pi = 0 \quad \text{on } \Gamma_i \cup \Gamma_b \cup \Gamma_s \cup \Gamma_w, \tag{33a}$$

$$u_1\lambda_2 + \frac{1}{Re}\frac{\partial \lambda_2}{\partial x_1} = -u_2 \quad \text{on } \Gamma_o, \tag{33b}$$

$$u_1\lambda_1 + \frac{1}{Re}\frac{\partial \lambda_1}{\partial x_1} = u_1 - u_{\text{parab}} + \Pi \quad \text{on } \Gamma_o. \tag{33c}$$

Finally, the total gradient of the cost objective with respect to the control is given by

$$\frac{\mathrm{d}\mathcal{J}(\mathbf{u})}{\mathrm{d}u_{\text{in}}} = \Pi(0,x_2) - \frac{1}{Re}\frac{\partial \lambda_1}{\partial x_1}(0,x_2), \tag{34}$$

where all values are evaluated at the inlet. For more details, the reader is invited to refer to [20].

We implement the DAL procedure in the icoFoam solver of OpenFOAM. The DAL optimal solution is obtained by iteratively solving the NS equations and their adjoint, updating the control $u_{\text{in}}(y)$ at each iteration with the gradient descent formula (16). The adjoint equations are solved with the same numerical methods as the direct equations. We choose the parabolic velocity profile $u_{\text{parab}}(y)$ as initial guess for the control $u_{\text{in}}(y)$ and employ a learning rate $\beta = 0.001$. The convergence of the cost objective during the DAL iterations is shown in Figure 9.

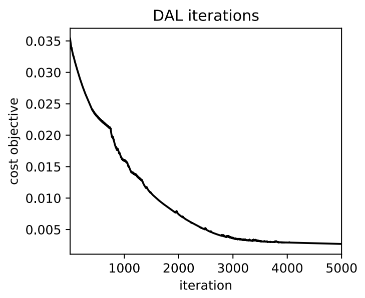

Figure 9: Cost objective during DAL iterations.