Equation identification for fluid flows via physics-informed neural networks

Alexander New [1] **Marisel Villafañe-Delgado** [1] **Charles Shugert** [1]

Abstract

Scientific machine learning (SciML) methods such as physics-informed neural networks (PINNs) are used to estimate parameters of interest from governing equations and small quantities of data. However, there has been little work in assessing how well PINNs perform for inverse problems across wide ranges of governing equations across the mathematical sciences. We present a new and challenging benchmark problem for inverse PINNs based on a parametric sweep of the 2D Burgers' equation with rotational flow. We show that a novel strategy that alternates between first- and second-order optimization proves superior to typical first-order strategies for estimating parameters. In addition, we propose a novel data-driven method to characterize PINN effectiveness in the inverse setting. PINNs' physics-informed regularization enables them to leverage small quantities of data more efficiently than the data-driven baseline. However, both PINNs and the baseline can fail to recover parameters for highly inviscid flows, motivating the need for further development of PINN methods.

1. Introduction

Physics-informed neural networks (PINNs) (Raissi et al., 2019) have emerged as a practical tool for the solution of partial differential equations (PDEs) in many scientific applications modeling complex systems. PINNs overcome some of the limitations of classical PDE solvers, improving on the expensive computational requirements of generating grids and developing bespoke solving schemes (Ma et al., 2021; Raissi et al., 2020; Ning et al., 2023; Haghighat et al., 2021; Alber et al., 2019; Cai et al., 2021; Hao et al., 2023).

PINNs can be used to solve both forward and inverse problems. For the PINN forward problem, a number of challenges and pathologies have been identified that arise when training PINNs, and mitigation methods have been suggested (Wang et al., 2021; 2022b;a; Krishnapriyan et al., 2021; New et al., 2023; Basir & Senocak, 2022; McClenny & Braga-Neto, 2020; Daw et al., 2022; Maddu et al., 2022; Lu et al., 2021b; Sukumar & Srivastava, 2022).

In contrast, the inverse problem has seen less systematic study, although some recent benchmark have included them (Hao et al., 2023). Inverse problems are of particular interest because, in practice, the PDEs modeling physical phenomena are often partially specified. Strategies to mitigate some of the challenges specific to the PINN inverse problem include improved optimization (Yu et al., 2022), new architectures (Aliakbari et al., 2023), and temporal methods for time-dependent PDEs (Mattey & Ghosh, 2022).

One challenge arising in the solution of inverse problems with PINNs is the limited range of evaluated PDEs. Studies typically consider a single PDE instance and do not vary parameters or other conditions. However, these parameters or constants (e.g. Reynolds number in fluid flow) that PDEs include may vary across application areas.

Additionally, the estimation of PDE parameters with PINNs presents particular challenges not encountered in the forward setting. While the forward setting requires fitting neural network (NN) parameters, in the inverse problem, the challenge is to fit both the NN and the PDE parameters, which differ significantly in their dimensionality. This results in limitations for methods such as Adam (Kingma & Ba, 2014) in the solution of inverse problems with PINNs.

Our contributions to the inverse problem for PINNs are three-fold. First, we propose a novel strategy for estimating PDE parameters using PINNs that alternates between using stochastic gradient descent (SGD) to update NN weights and using Newton's method to estimate PDE parameters. Second, we introduce a 2D Burgers' equation benchmark problem with varying parameter coefficients across 10 solutions, including highly viscous and inviscid flows. Third, we propose an approach for estimating the benefit of using physics-informed regularization in PDE estimation problems. We show that the PINNs' use of physics-based regularization enhances its effectiveness in parameter estimation problems, although PINNs can still struggle to simultane-

[1]Johns Hopkins University Applied Physics Laboratory, 11100 Johns Hopkins Rd, Laurel, MD 20723, USA. Correspondence to: Alexander New <alex.new@jhuapl.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

ously minimize loss criteria involving PDEs and data.

The rest of this paper is organized as follows: In Sections 2.1 to 2.3, we introduce the PDE inverse problem and explain how it applies to the 2D Burgers' equation, and in Section 2.4, we present an overview of PINNs. Section 2.5, Section 2.6, and Section 2.7 describe our main contributions in this work, highlighting the unique challenges that occur in the PINN inverse problem. Finally, in Sections 3.1 to 3.3, we evaluate methods on the 2D Burgers' equation.

2. Methods

2.1. Problem formulation

We consider a function $\mathbf{u} : \Omega \to \mathbb{R}^m$ defined on $\Omega \subseteq \mathbb{R}^n$. The function $\mathbf{u}$ satisfies a set of differential operators $N_i(\cdot;\phi)$ for $i \in I_N$. The domain Ω has r boundaries Γ_j, $j = 1, \ldots, r$ (including, possibly, the beginning of a temporal dimension). The function $\mathbf{u}$ satisfies boundary condition (BC) or initial condition (IC) operators $B_j(\cdot;\phi)$ on boundaries Γ_j, for a set of indices $j \in I_b$. Here, ϕ is one or more scalar parameters defining the operators N_i and B_j. Then the PDE may be written as:

$$\begin{aligned} N_i(\mathbf{u};\phi)(\mathbf{x}) &= 0 \quad \mathbf{x} \in \Omega, \quad i \in I_N \\ B_j(\mathbf{u};\phi)(\mathbf{x}) &= 0 \quad \mathbf{x} \in \Gamma_j, \; j \in I_B. \end{aligned} \tag{1}$$

The type of BC can include Dirichlet BCs ($B_j(\mathbf{u};\phi)(\mathbf{x}) = \mathbf{u}(\mathbf{x}) - \mathbf{f}_j(\mathbf{x};\phi)$ for a function $\mathbf{f}_j$), Neumann BCs ($B_j(\mathbf{u};\phi)(\mathbf{x}) = (\hat{\mathbf{n}} \cdot \nabla \mathbf{u})(\mathbf{x}) - \mathbf{g}_j(\mathbf{x};\phi)$ for a function $\mathbf{g}_j$), periodic BCs, or another type. The differential operators N_i might be linear or nonlinear. A BC could be parameterized by the speed of an inflow BC (Collins et al., 2023), or the domain Ω could be parameterized by the size and shape of a defect in its interior (Zhang et al., 2023).

Given samples $\mathcal{D} = \{(\mathbf{u}_n, \mathbf{x}_n)\}_{n=1}^N$ from the solution, the inverse problem's goal is to (i) obtain a function that can fill in $\mathbf{u}$ for the remainder of the domain, and (ii) identify values for ϕ consistent with the training data and BCs.

2.2. The 2D Burgers' equation

In this work, we focus on the sourceless time-varying Burgers' equation. It has two solution components ($\mathbf{u} = (u, v)$), two spatial inputs and one temporal input ($\mathbf{x} = (x, y, t)$) defined on the domain $[0, 1] \times [0, 1] \times [0, 0.5]$:

$$\mathbf{u}_t + \alpha\, \mathbf{u} \cdot \nabla \mathbf{u} - \nu \Delta \mathbf{u} = 0 \tag{2}$$

$$\begin{aligned} u(x, y, 0) &= \sin(6\pi y) x(1 - x) \\ v(x, y, 0) &= -\sin(6\pi x) y(1 - y) \end{aligned} \tag{3}$$

$$\begin{aligned} \mathbf{u}(x, 0, t) = \mathbf{u}(x, 1, t) &= 0 \\ \mathbf{u}(0, y, t) = \mathbf{u}(1, y, t) &= 0. \end{aligned} \tag{4}$$

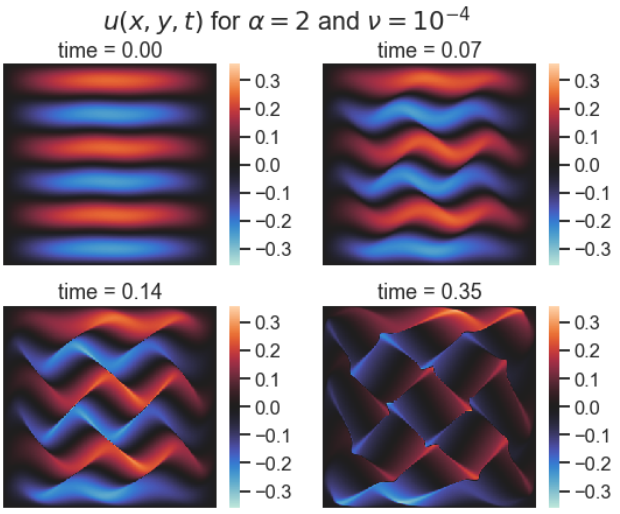

Figure 1. We show a few temporal snapshots of the u-component of a Burgers' equation solution. At $t = 0$, the solution exhibits variation only in the y-direction; as time progresses (e.g., for $t = 0.14$ and $t = 0.35$), rotational flow develops.

The parameters $\phi = \{\alpha, \nu\}$ characterize the convection α and diffusion ν components of the solution. When $\alpha = 0$, eq. 2 reduces to a series of uncoupled heat equations. When $\nu = 0$, eq. 2 reduces to an inviscid Burgers' equation.

Eq. 2 can be reparameterized into the usual Burgers' equation by rescaling the domain with the variable transformation $\mathbf{x} \to \alpha\, \mathbf{x}$:

$$\mathbf{u}_t + \alpha\, \mathbf{u} \cdot \nabla \mathbf{u} - \nu \Delta \mathbf{u} \quad \to \quad \mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u} - \frac{\nu}{\alpha^2} \Delta \mathbf{u}.$$

After this reparameterization, the Cole-Hopf transformation (Cole, 1951; Hopf, 1950) can be used to analytically solve the Burgers' equation, so long as the solution does not admit irrotational flow (i.e., if $\nabla \times \mathbf{u} = 0$). Thus, we choose ICs that vanish at the boundaries but have nonzero curl. In this way, the data we generate is an example of the simplest nontrivial (rotational) Burgers' flow.

2.3. Existing uses of PINNs for Burgers' equation

To the best of our knowledge, there has not been a previous study of the 2D Burgers' equation that uses two solution components and varies its parameters. In general, the 1D Burgers' equation is a common benchmark problem for forward PINNs Ren et al. (2022); Carniello et al. (2022); Mathias et al. (2022); Rosofsky et al. (2023), but there has been less work studying the 2D Burgers' equation, especially in the inverse problem setting for varying parameters Alkhadhr & Almekkawy (2021); Xu et al. (2023).

In the forward setting, Ren et al. (2022) proposed a convolutional-recurrent architecture and evaluated its per-

formance in an extrapolation setting on the 2D Burgers' problem with two components and fixed viscosity. Carniello et al. (2022) proposed a PINN to solve the scalar 2D inviscid Burgers' with data from a Riemann problem, which exhibits discontinuities and refraction and shock waves. Mathias et al. (2022) proposed a data augmentation strategy and evaluated its performance in the 2D Burgers' forward problem with two components. Kim et al. (2022) proposed a nonlinear manifold reduced order model (ROM) for estimating the two components of the viscous Burgers' with large Reynolds numbers. Rosofsky et al. (2023) evaluated various settings of the Burgers' equation in 1D and 2D: 2D scalar, 2D inviscid, 2D vector Burgers' using a physics-informed neural operator. In the inverse problem setting, Alkhadhr & Almekkawy (2021) proposed PINNs to solve the 1D and 2D Burgers' in the forward and inverse problems, however, their solution has a single component, and they only consider a single set of PDE parameters. Xu et al. (2023) also solved the 2D Burgers' inverse problem for equation discovery, but with one component.

2.4. Physics-informed neural networks

In the inverse problem PINN approach (Raissi et al., 2019), a neural network $\mathbf{u}_\theta$ parameterized by weights θ is trained to satisfy both the PDE and the data:

$$\theta^*, \phi^* = \arg\min_{\theta} L(\theta; \phi), \tag{5}$$

where

$$\begin{aligned} L(\theta;\phi) &= L_{PDE}(\theta;\phi) + L_{BC}(\theta;\phi) \\ &\quad + L_{Data}(\theta) \qquad (6) \\ L_{PDE}(\theta;\phi) &= \sum_{i\in I_N} \lambda_i \sum_{\mathbf{x}_\Omega \in \Omega} ||N_i(\mathbf{u}_\theta;\phi)(\mathbf{x}_\Omega)||^2 \qquad (7) \\ L_{BC}(\theta) &= \sum_{j\in I_B} \lambda_j \sum_{\mathbf{x}_j \in \Gamma_j} ||B_j(\mathbf{u}_\theta;\phi)(\mathbf{x}_j)||^2 \qquad (8) \\ L_{Data}(\theta) &= \lambda_{Data} \sum_{n=1}^{N} ||\mathbf{u}_n - \mathbf{u}_\theta(\mathbf{x}_n)||^2. \qquad (9) \end{aligned}$$

Here, λ_i, λ_j, and λ_{Data} are nonnegative weights for each component of the loss function, $\{\mathbf{x}_\Omega\}$ is a set of points sampled from the domain Ω, and $\{\mathbf{x}_j\}$ is a set of points sampled from each boundary Γ_j. The set of points can either be chosen at the start of training (via random sampling or a chosen discretization), or it can change or grow during training (Daw et al., 2022; Lu et al., 2021a).

A noted challenge in training PINNs is that their loss function can contain many components that must all be minimized for the problem to be solved. To reduce the complexity of the PINN training process and focus on the underlying inverse problem, we use an NN that vanishes by construction at the domain's boundaries:

$$\begin{aligned} u_\theta(x,y,t) &= x(1-x)y(1-y)\mathrm{NN}_u(x,y,t;\theta) \\ v_\theta(x,y,t) &= x(1-x)y(1-y)\mathrm{NN}_v(x,y,t;\theta). \end{aligned} \tag{10}$$

Here NN is a multi-layer perceptron (MLP) or encoder-decoder multi-layer perceptron (ED-MLP) (Wang et al., 2021) with two output nodes (NN_u and NN_v). Similar strategies use periodic MLPs (Dong & Ni, 2021; Wang et al., 2022a) when solving PDEs on periodic domains.

2.5. Estimating PDE parameters using neural networks

In the forward problem, PINNs are often trained with a combination of first-order optimization methods like SGD combined with Adam (Kingma & Ba, 2014) and approximate second-order methods like L-BFGS (Liu & Nocedal, 1989). However, estimating PDE parameters from a partially-specified set of governing equations and some amount of data differs in several key respects from the typical training done to fit NN parameters.

In particular, ϕ's dimensionality is much lower than θ's. Here, ϕ has only two components, while the typical NN may have between 10^5 and 10^7 weights. Furthermore, the scale of the components of ϕ has a physical meaning based on the governing equations and domain size. These factors combine to make techniques like Adam (Kingma & Ba, 2014) or MultiAdam (Yao et al., 2023) inefficient for estimating PDE parameters using PINNs. For example, misspecified step sizes for ϕ can cause its iterates to diverge.

In this work, we use Newton's method to estimate ϕ but rely on Adam for fitting θ. To motivate this, observe that, for the Burgers' equation, each governing equation (eq. 2) is a linear function of its parameters $\phi = \{\alpha, \nu\}$. This makes the PINN loss (eq. 6) convex and quadratic in ϕ, meaning that, for fixed θ, a single Newton step yields the optimal ϕ.[1] To the best of our knowledge, this observation has not been widely discussed in the inverse PINN literature.

2.6. Data-driven PDE parameter estimation

An understudied challenge in the use of PINNs is the question of how much labeled data is needed to reconstruct the full solution and accurately estimate the PDE parameters. For example, the Poisson and diffusion inverse problems studied in (Hao et al., 2023) use a fixed labeled dataset size of 2500 points. Especially for problems involving nonlinear PDEs, existing theory may not be sufficient to give conditions for a problem to be identifiable.

Thus, here we propose a novel baseline for assessing the data efficiency benefits gained by using PINNs for inverse

[1] It is not generally the case that the PINN learning problem will be linear in its parameters.

Algorithm 1 Estimating PDE parameters using a data-driven NN

Require: Sample of PDE solutions $\{(\mathbf{u}_n, \mathbf{x}_n)\}_n^N$
Require: Differential operators N_i and boundary condition operators B_j
Require: Neural network architecture $\mathbf{u}_\theta$
1: Minimize $\hat{\theta} = \arg\min_\theta L_{BC}(\theta) + L_{Data}(\theta)$ with SGD
2: Minimize $\hat{\phi} = \arg\min_\phi L_{PDE}(\hat{\theta}; \phi)$ with Newton's method
3: Return PDE parameter estimates $\hat{\phi}$

problems. It is summarized in Algorithm 1. For a fixed sample of labeled data, we train a PINN to minimize the sum of the BC loss (eq. 8) and the data loss (eq. 9). We then use Newton's method to estimate the optimal PDE parameters that minimize the PDE residual (eq. 7) while holding the NN weights constant. Algorithm 1 relies on the BC loss L_{BC} being independent of the parameters ϕ. If this were not the case, then the first step would train the NN using only L_{data}.

For a sufficiently large quantity of labeled data and a sufficiently expressive NN, we expect the NN to interpolate the full discretized solution. If the discretization error is not too high, this should enable the parameters ϕ to be accurately recovered. The benefit of a PINN may be observed in the data quantity regime where a physics-unaware NN cannot fit the entire solution, but a PINN can.

2.7. Data generation

ν	α
10^{-4}	$1.0, 1.5, 2.0$
10^{-3}	$1.0, 1.5, 2.0$
10^{-2}	$0.5, 1.0, 1.5, 2.0$

Table 1. The parameter values of the Burgers' equation (eq. 2) used to generate ground truth solutions.

Solutions to the Burgers' equation were obtained by using FiPy: A Finite Volume PDE Solver (Guyer et al., 2009). We use a 256×256 uniform discretization of $[0, 1] \times [0, 1]$ for the spatial domain. There are 72 timesteps that cover $t = 0$ to $t = 0.499$, which means that solutions have $256 \cdot 256 \cdot 72 = 4{,}718{,}592$ points total per simulation.

The diffusive term in the Burgers' equation was handled using an implicit scheme. To handle the nonlinear convective term, the Burgers' Equation was solved in its conservative form. The convective $\mathbf{u} \cdot \nabla \mathbf{u}$ terms were solved using an implicit discretization scheme, and the $\Delta \mathbf{u}$ terms were handled using a Power-Law discretization scheme (Versteeg & Malalasekera, 2007). At each time step, the PDE used a linear LU solver with an Algebraic Multigrid Preconditioner (Bell et al., 2023).

Using the parameter values in Table 1, we generate 10 ground truth solutions to the Burgers' equation. We view α/ν as an effective-order parameter for the system's Reynolds number. By varying ν logarithmically, we study how effective the PINNs are in both highly viscous and highly inviscid regimes. In contrast, we vary α linearly to study how sensitive PINNs are at learning parameters that have smaller effects on the governing fluid equations.

3. Results

3.1. Implementation and evaluation procedure

For each PDE parameter value and amount of training data (2048, 8192, 32768 points), we train PINNs using both Adam and Newton's method for parameter value estimation. Similarly, for each PDE parameter and amount of training data (128, 512, 2048, 8192, 32768, 131072 points), we apply Algorithm 1 for data-driven parameter estimation.

We implement PINNs with `pinn-jax` (New et al., 2023)[2], which uses `jax` (Bradbury et al., 2018), `flax` (Heek et al., 2023), and `optax` (Babuschkin et al., 2020). PDE derivatives are calculated with the forward-mode `jacfwd` function (Baydin et al., 2017). All computations use double precision. See Appendix A for details on hyperparameters. For training PINNs, we use Adam (Kingma & Ba, 2014).

We initialize the convection coefficient α by sampling from $[0, 5]$ uniformly, and we initialize the diffusion coefficient ν by sampling from $[0, 0.5]$ uniformly. This choice is analogous to, in Bayesian methods for inverse problems, imposing a uniform prior on the unknown parameters (e.g., (Christopher et al., 2018; Doronina et al., 2020)).

For a ground truth solution $\mathbf{u}$, a trained solution $\mathbf{u}_\theta$, and a set of points $\mathbf{X} \subseteq \Omega$, we evaluate model predictions with the relative error E_{rel}:

$$E_{rel}(\mathbf{u}, \mathbf{u}_\theta; \mathbf{X}) = \frac{\left(\sum_{\mathbf{x} \in \mathbf{X}} ||\mathbf{u}(\mathbf{x}) - \mathbf{u}_\theta(\mathbf{x})||^2\right)^{1/2}}{\left(\sum_{\mathbf{x} \in \mathbf{X}} ||\mathbf{u}(\mathbf{x})||^2\right)^{1/2}}, \quad (11)$$

where $\mathbf{X}$ is a uniform discretization of the domain and its boundaries. For reporting error in estimating α, we use the scalar relative error $|\hat{\alpha} - \alpha|/\alpha$. Because ν takes values across different orders of magnitudes, we report relative error of logs: $|\log_{10}(\hat{\nu} + \epsilon) - \log_{10}(\nu + \epsilon)|/|\log_{10}(\nu + \epsilon)|$, where $\epsilon = 10^{-8}$ prevents numerical overflow in the case that $\hat{\nu} = 0$. We follow Hao et al. (2023) and consider a relative error that is 10% or less to be sufficiently accurate for the estimation problem to be solved.

For PINNs, our results are from the model checkpoint that attained the lowest loss (eq. 6). For the data-driven estima-

[2] https://github.com/newalexander/pinn-jax

tion, our results are from the model checkpoint that attained the lowest error on the data (eq. 9).

3.2. Methods assessment

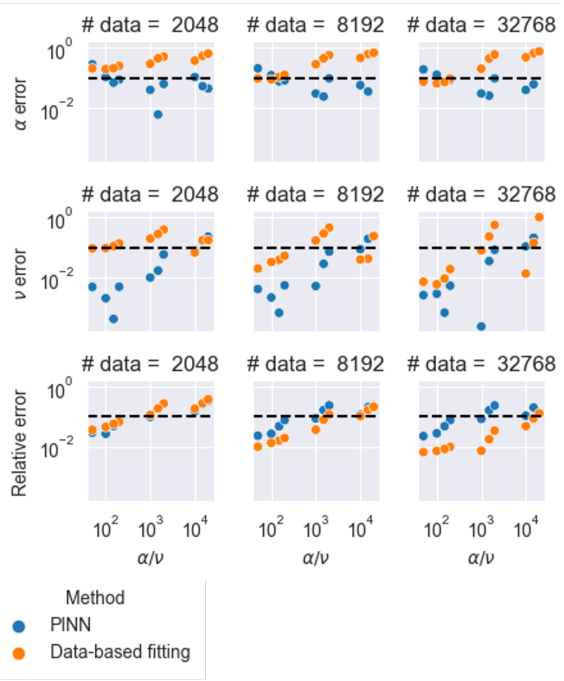

Figure 2. For given quantities of data, we compare the estimation accuracy of PINNs (using Newton's method) and the data-driven strategy (Algorithm 1). Needing to minimize both the PDE residual and data losses makes PINNs less effective at fitting the solution directly, yielding typically higher relative errors. However, the PINNs are generally better at fitting the PDE parameters α and ν.

In Figure 2, we summarize the results for PINNs and the data-driven strategy in Figure 2, where $2048, 8192$, or 32768 training points were supplied.

PINNs are trained using a multi-objective loss function based on data and PDEs. This can make their convergence noisy and each criteria difficult to simultaneously satisfy. Thus, the data-driven approach can be better at fitting the supplied training data and attaining low relative error on the full solution.

However, fitting only the data to between 1% and 10% relative error generally does not guarantee that the PDE parameters are estimated to within 10% error or less. In contrast, PINNs are able to more consistently estimate ν and α given the amount of data provided. PINN parameter estimation can be successful with only 8192 training points, i.e., approximately 0.2% of the solution.

Both methods are less accurate in the regime where α/ν is large. This is unsurprising, as the fluid flow changes rapidly across temporal and spatial scales in that parameter regime. This echoes work in PINN forward modeling highlighting that model predictions can be inaccurate as parameters driving system complexity vary (Krishnapriyan et al., 2021; New et al., 2023).

3.3. Analysis of results

Figure 3 contains further details about the performance of the data-driven estimation strategy, in the settings where 8192 or more data points are supplied. Results with fewer training points are in Figures 6 and 7, and Figure 8 in Appendix B. In Figure 4, as well as in Figure 5 in Appendix B, we show further results in terms of parameter estimation and accuracy for the PINNs.

With low amounts of labeled data, Algorithm 1 unsurprisingly performs poorly, but even in high-data settings (e.g., 131072 training points, ten times as many as the PINNs have), parameter estimation often still fails. We note that the largest number of supplied training points we use, 131072, is still less than 3% of the total number of points in the solution. Training on larger datasets could enable more reliable parameter estimation.

Although the PINNs are more accurate than Algorithm 1, similar trends to Figure 3 hold, namely that performance degrades in the regime where α/ν is large. Increasingly, increasing the amount of data supplied does not consistently improve PINN accuracy, in comparison to Algorithm 1. This is likely a consequence of the underlying complexity and ill-conditionedness of the PINN optimization problem.

PDE optimizer	# data	α error	Relative error	ν error
Adam	2048	0.080	0.152	0.093
Newton	2048	0.086	0.150	0.056
Adam	8192	0.085	0.139	0.097
Newton	8192	0.085	0.130	0.065
Adam	32768	0.067	0.139	0.101
Newton	32768	0.088	0.126	0.071

Table 2. We show relative errors and estimation errors for PINNs, averaged over the parameter space, comparing using Adam and Newton's method for the PDE estimation component. Newton's enables better estimation of ν and better fitting of the solution, while Adam yields superior estimation of α.

4. Conclusion

We have presented a novel PINNs benchmark for the vector 2D Burgers' equation with varying equation parameters in the inverse problem setting. Our strategy combines SGD and Newton's method to learn the NN and PDE parameters, respectively. This is a first step towards completion of a major gap in the PINNs literature. First we demonstrated the recovery of PDE parameters in inverse problems across several viscous and inviscid flow conditions. Second, we

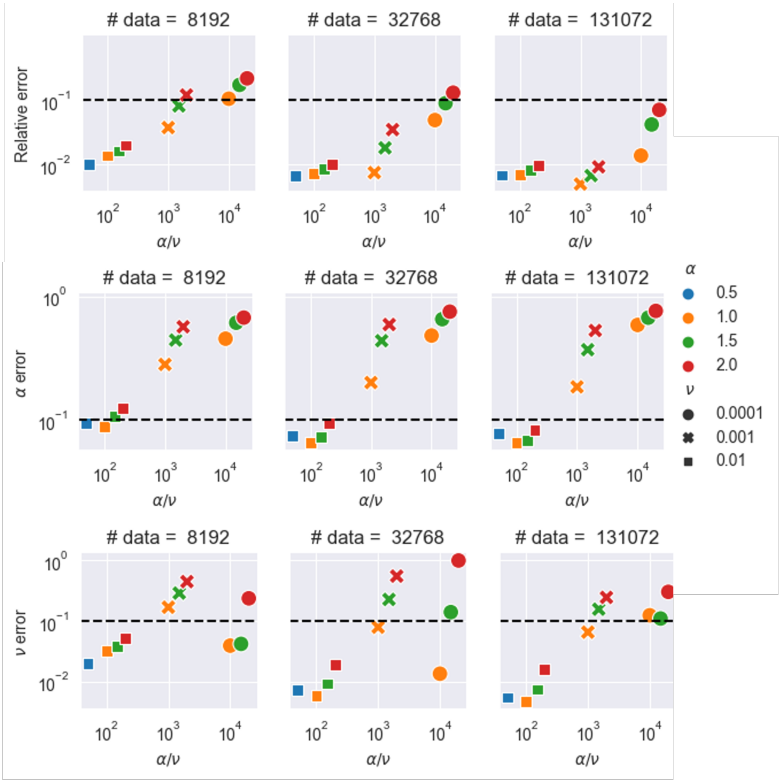

Figure 3. For varying amounts of training data, we plot the relative errors in estimating the Burgers' solution (top), convection coefficient α (middle) and diffusion coefficient ν (bottom), using the data-driven NN (Algorithm 1) strategy across different Burgers' parameters. The black dashed line indicates 10% or less error, our threshold for success. With 131072 points, the NN can achieve less than 10% solution relative error for every parameter configuration. However, even with 131072 data points, it struggles to estimate parameters, failing at estimating α for every configuration other than $\nu = 0.01$ and failing at estimating ν in five out of the ten parameter configurations.

addressed the limitations of optimizers such as Adam in estimating PDE parameters via the use of Newton's method.

As future work, we advocate for further development of challenging inverse problems, including those that include partially-known BCs or ICs (Mattey & Ghosh, 2022), or systematic application of noise to data solution data (e.g., (Hao et al., 2023)). For these and more challenging settings, we expect that additional training strategies should be employed, such as improved sampling (Wang et al., 2024).

In the forward problem PINN literature, breaking time domains into smaller subsets and training PINNs sequentially on each subset is a common strategy (Krishnapriyan et al., 2021; Wang et al., 2022a), and predicting phenomena across large time domains is a known challenge for PINNs (Meng et al., 2020; Wang et al., 2022a; Daw et al., 2022). Here, we estimated parameters from solutions confined to $t \in [0, 0.499]$, but using either a smaller or large time domain could have enabled generally better parameter estimation. Thus, we suggest deeper exploration into the impact of time domain size on the parameter estimation problem, especially when the data feature discontinuities or sudden changes.

Acknowledgments

This work was supported by internal research and development funding from the Research and Exploratory Development Mission Area of the Johns Hopkins University Applied Physics Laboratory.

References

Alber, M., Buganza Tepole, A., Cannon, W. R., De, S., Dura-Bernal, S., Garikipati, K., Karniadakis, G., Lytton, W. W., Perdikaris, P., Petzold, L., and Kuhl, E. Integrating machine learning and multiscale modeling—perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences. *npj Digital Medicine*, 2(1):115, Nov 2019. ISSN 2398-6352. doi: 10.1038/s41746-019-0193-y. URL https://doi.org/10.1038/s41746-019-0193-y.

Aliakbari, M., Soltany Sadrabadi, M., Vadasz, P., and Arzani, A. Ensemble physics informed neural networks:

Figure 4. For varying amounts of training data (columns) and different PDE optimizers (rows), we plot the relative errors in estimating the diffusion coefficient ν (left) and convection coefficient α (right), using PINNs. The black dashed line indicates 10% or less error, our threshold for success. Compared to the data-driven baseline (Figure 3), PINNs are more successful in recovering parameters at a given quantity of labeled training data. They can correctly estimate ν across values of α except when $\nu = 0.0001$ (i.e., when the fluid is highly inviscid). They are also successful in estimating α, except in the $\nu = 0.01$ regime.

A framework to improve inverse transport modeling in heterogeneous domains. *Physics of Fluids*, 35(5), 2023.

Alkhadhr, S. and Almekkawy, M. A combination of deep neural networks and physics to solve the inverse problem of burger's equation. In *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 4465–4468, 2021. doi: 10.1109/EMBC46164.2021.9630259.

Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., Buchlovsky, P., Budden, D., Cai, T., Clark, A., Danihelka, I., Dedieu, A., Fantacci, C., Godwin, J., Jones, C., Hemsley, R., Hennigan, T., Hessel, M., Hou, S., Kapturowski, S., Keck, T., Kemaev, I., King, M., Kunesch, M., Martens, L., Merzic, H., Mikulik, V., Norman, T., Papamakarios, G., Quan, J., Ring, R., Ruiz, F., Sanchez, A., Schneider, R., Sezener, E., Spencer, S., Srinivasan, S., Stokowiec, W., Wang, L., Zhou, G., and Viola, F. The DeepMind JAX Ecosystem, 2020. URL http://github.com/deepmind.

Basir, S. and Senocak, I. Physics and equality constrained artificial neural networks: Application to forward and inverse problems with multi-fidelity data fusion. *Journal of Computational Physics*, 463:111301, 2022. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2022.111301. URL https://www.sciencedirect.com/science/article/pii/S0021999122003631.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. Automatic Differentiation in Machine Learning: A Survey. *J. Mach. Learn. Res.*, 18(1):5595–5637, Jan 2017. ISSN 1532-4435.

Bell, N., Olson, L. N., Schroder, J., and Southworth, B. PyAMG: Algebraic multigrid solvers in python. *Journal of Open Source Software*, 8(87):5495, 2023. doi: 10.21105/joss.05495. URL https://doi.org/10.21105/joss.05495.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Cai, S., Wang, Z., Wang, S., Perdikaris, P., and Karniadakis, G. E. Physics-Informed Neural Networks for Heat Transfer Problems. *Journal of Heat Transfer*, 143(6), 04 2021. ISSN 0022-1481. doi: 10.1115/1.4050542. URL https://doi.org/10.1115/1.4050542. 060801.

Carniello, R., Florindo, J. B., and Abreau, E. A PINN computational study for a scalar 2d inviscid Burgers model with Riemann data. In *Manuscrito de conferência em revisao*, 2022.

Christopher, J. D., Wimer, N. T., Lapointe, C., Hayden, T. R. S., Grooms, I., Rieker, G. B., and Hamlington, P. E. Parameter estimation for complex thermal-fluid flows using approximate bayesian computation. *Phys. Rev. Fluids*, 3:104602, Oct 2018. doi: 10.1103/PhysRevFluids.3.104602. URL https://link.aps.org/doi/10.1103/PhysRevFluids.3.104602.

Cole, J. D. On a quasi-linear parabolic equation occurring in aerodynamics. *Quarterly of Applied Mathematics*, 9:225–236, 1951. URL https://api.semanticscholar.org/CorpusID:39662248.

Collins, G., New, A., Darragh, R. A., Damit, B. E., and Stiles, C. D. Rapid prediction of two-dimensional airflow in an operating room using scientific machine learning. In *NeurIPS 2023 AI for Science Workshop*, 2023. URL https://openreview.net/forum?id=mUQrw0rIZN.

Daw, A., Bu, J., Wang, S., Perdikaris, P., and Karpatne, A. Mitigating propagation failures in pinns using evolutionary sampling, 2022.

Dong, S. and Ni, N. A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks. *Journal of Computational Physics*, 435:110242, 2021. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2021.110242. URL https://www.sciencedirect.com/science/article/pii/S0021999121001376.

Doronina, O. A., Murman, S. M., and Hamlington, P. E. Parameter estimation for RANS models using approximate bayesian computation, 2020.

Guyer, J. E., Wheeler, D., and Warren, J. A. Fipy: Partial differential equations with python. *Computing in Science & Engineering*, 11(3):6–15, 2009. doi: 10.1109/MCSE.2009.52.

Haghighat, E., Raissi, M., Moure, A., Gomez, H., and Juanes, R. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379:113741, 2021. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2021.113741. URL https://www.sciencedirect.com/science/article/pii/S0045782521000773.

Hao, Z., Yao, J., Su, C., Su, H., Wang, Z., Lu, F., Xia, Z., Zhang, Y., Liu, S., Lu, L., and Zhu, J. Pinnacle: A comprehensive benchmark of physics-informed neural networks for solving pdes, 2023.

Heek, J., Levskaya, A., Oliver, A., Ritter, M., Rondepierre, B., Steiner, A., and van Zee, M. Flax: A neural network library and ecosystem for JAX, 2023. URL http://github.com/google/flax.

Hopf, E. The partial differential equation ut + uux = μxx. *Communications on Pure and Applied Mathematics*, 3:201–230, 1950. URL https://api.semanticscholar.org/CorpusID:121837938.

Kim, Y., Choi, Y., Widemann, D., and Zohdi, T. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *Journal of Computational Physics*, 451:110841, 2022.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization, 2014. doi:10.48550/ARXIV.1412.6980.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. Characterizing Possible Failure Modes in Physics-Informed Neural Networks. *Advances Neural Inf. Process. Syst.*, 34, 2021.

Liu, D. C. and Nocedal, J. On the Limited Memory BFGS Method for Large Scale Optimization. *Mathematical Programming*, 45(1):503–528, Aug 1989. ISSN 1436-4646. doi: 10.1007/BF01589116.

Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. DeepXDE: A Deep Learning Library for Solving Differential Equations. *SIAM Review*, 63(1):208–228, 2021a.

Lu, L., Pestourie, R., Yao, W., Wang, Z., Verdugo, F., and Johnson, S. G. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021b. doi: 10.1137/21M1397908. URL https://doi.org/10.1137/21M1397908.

Ma, W., Liu, Z., Kudyshev, Z. A., Boltasseva, A., Cai, W., and Liu, Y. Deep learning for the design of photonic structures. *Nature Photonics*, 15(2):77–90, Feb 2021. ISSN 1749-4893. doi: 10.1038/s41566-020-0685-y. URL https://doi.org/10.1038/s41566-020-0685-y.

Maddu, S., Sturm, D., Müller, C. L., and Sbalzarini, I. F. Inverse Dirichlet Weighting Enables Reliable Training of Physics Informed Neural Networks. *Machine Learning: Science and Technology*, 3(1):015026, feb 2022. doi: 10.1088/2632-2153/ac3712.

Mathias, M. S., de Almeida, W. P., Coelho, J. F., de Freitas, L. P., Moreno, F. M., Netto, C. F., Cozman, F. G., Reali Costa, A. H., Tannuri, E. A., Gomi, E. S., et al. Augmenting a physics-informed neural network for the 2d burgers equation by addition of solution data points. In *Brazilian Conference on Intelligent Systems*, pp. 388–401. Springer, 2022.

Mattey, R. and Ghosh, S. A novel sequential method to train physics informed neural networks for allen cahn and cahn hilliard equations. *Computer Methods in Applied Mechanics and Engineering*, 390:114474, 2022.

McClenny, L. and Braga-Neto, U. Self-Adaptive Physics-Informed Neural Networks using a Soft Attention Mechanism, 2020. doi:10.48550/ARXIV.2009.04544.

Meng, X., Li, Z., Zhang, D., and Karniadakis, G. E. PPINN: Parareal Physics-Informed Neural Network for Time-Dependent PDEs. *Comput. Methods Appl. Mechanics Eng.*, 370:113250, 2020. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2020.113250.

New, A., Eng, B., Timm, A. C., and Gearhart, A. S. Tunable complexity benchmarks for evaluating physics-informed neural networks on coupled ordinary differential equations. In *2023 57th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–8, 2023. doi: 10.1109/CISS56502.2023.10089728.

Ning, L., Cai, Z., Dong, H., Liu, Y., and Wang, W. A peridynamic-informed neural network for continuum elastic displacement characterization. *Computer Methods in Applied Mechanics and Engineering*, 407:115909, 2023. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2023.115909. URL https://www.sciencedirect.com/science/article/pii/S0045782523000324.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. *J. Comp. Phys.*, 378: 686–707, 2019.

Raissi, M., Yazdani, A., and Karniadakis, G. E. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367 (6481):1026–1030, 2020. doi: 10.1126/science.aaw4741. URL https://www.science.org/doi/abs/10.1126/science.aaw4741.

Ren, P., Rao, C., Liu, Y., Wang, J.-X., and Sun, H. Phycrnet: Physics-informed convolutional-recurrent network for solving spatiotemporal pdes. *Computer Methods in Applied Mechanics and Engineering*, 389:114399, 2022.

Rosofsky, S. G., Al Majed, H., and Huerta, E. Applications of physics informed neural operators. *Machine Learning: Science and Technology*, 4(2):025022, 2023.

Sukumar, N. and Srivastava, A. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 389:114333, 2022. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2021.114333. URL https://www.sciencedirect.com/science/article/pii/S0045782521006186.

Versteeg, H. K. and Malalasekera, W. *An introduction to computational fluid dynamics: the finite volume method*. Pearson education, 2007.

Wang, S., Teng, Y., and Perdikaris, P. Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

Wang, S., Sankaran, S., and Perdikaris, P. Respecting Causality is all you Need for Training Physics-Informed Neural Networks, 2022a. doi:10.48550/ARXIV.2203.07404.

Wang, S., Yu, X., and Perdikaris, P. When and Why PINNs Fail to Train: A Neural Tangent Kernel Perspective. *Journal of Computational Physics*, 449:110768, 2022b.

Wang, S., Li, B., Chen, Y., and Perdikaris, P. Piratenets: Physics-informed deep learning with residual adaptive networks. *arXiv preprint arXiv:2402.00326*, 2024.

Xu, H., Zeng, J., and Zhang, D. Discovery of partial differential equations from highly noisy and sparse data with physics-informed information criterion. *Research*, 6:0147, 2023.

Yao, J., Su, C., Hao, Z., Liu, S., Su, H., and Zhu, J. MultiAdam: Parameter-wise scale-invariant optimizer for multiscale training of physics-informed neural networks. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 39702–39721. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/yao23c.html.

Yu, J., Lu, L., Meng, X., and Karniadakis, G. E. Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. *Computer Methods in Applied Mechanics and Engineering*, 393:114823, 2022.

Zhang, X., Wang, L., Helwig, J., Luo, Y., Fu, C., Xie, Y., Liu, M., Lin, Y., Xu, Z., Yan, K., Adams, K., Weiler, M., Li, X., Fu, T., Wang, Y., Yu, H., Xie, Y., Fu, X., Strasser, A., Xu, S., Liu, Y., Du, Y., Saxton, A., Ling, H., Lawrence, H., Stärk, H., Gui, S., Edwards, C., Gao, N., Ladera, A., Wu, T., Hofgard, E. F., Tehrani, A. M., Wang, R., Daigavane, A., Bohde, M., Kurtin, J., Huang, Q., Phung, T., Xu, M., Joshi, C. K., Mathis, S. V., Azizzadenesheli, K., Fang, A., Aspuru-Guzik, A., Bekkers, E., Bronstein, M., Zitnik, M., Anandkumar, A., Ermon, S., Liò, P., Yu, R., Günnemann, S., Leskovec, J., Ji, H., Sun, J., Barzilay, R., Jaakkola, T., Coley, C. W., Qian, X., Qian, X., Smidt, T., and Ji, S. Artificial intelligence for science in quantum, atomistic, and continuum systems, 2023.

A. Hyperparameters

Hyperparameter	Value
Number of hidden units	256
Number of layers	10
Activation function	$\tanh$
Batch size (Data loss)	2048
Batch size (IC loss)	2048
Batch size (PDE loss)	8192
λ_{Data}	1
λ_{IC}	1
Optimizer	Adam
Number of epochs	50000
Initial learning rate	$5 \cdot 10^{-3}$
Minimum learning rate	10^{-6}
Exponential decay rate	0.925
Exponential decay interval	5000

Table 3. Hyperparameters used for the data-driven estimation strategy (Algorithm 1). For the data loss, we use a batch size of the minimum of the number of labeled data points and 2048. Models use the ED-MLP (Wang et al., 2021), modified to exactly satisfy the BCs (eqs. 10).

Hyperparameter	Value
Number of hidden units	256
Number of layers	10
Activation function	$\tanh$
Batch size (Data loss)	Varies
Batch size (IC loss)	1024
Batch size (PDE loss)	2048
λ_{PDE}	1
λ_{Data}	10
λ_{IC}	10
NN Optimizer	Adam
PDE Optimizer	{Adam, Newton}
Number of gradient steps	100000
Parameter estimation interval	$\{10\,(\text{Adam}), 100\,(\text{Newton})\}$
Initial NN learning rate	$5 \cdot 10^{-3}$
Minimum NN learning rate	10^{-5}
Exponential decay rate	0.925
Exponential decay interval	5000
PDE parameter step size	10^{-3}

Table 4. Hyperparameters used for PINNs. For the data loss, we use all available labeled data points for each gradient update. Models use the ED-MLP (Wang et al., 2021), modified to exactly satisfy the BCs (eqs. 10). When training, we only update the PDE parameters every n epochs, where $n = 10$ if the PDE optimizer is Adam, and $n = 100$ if the PDE parameter optimizer is Newton's method.

B. Supplementary Figures

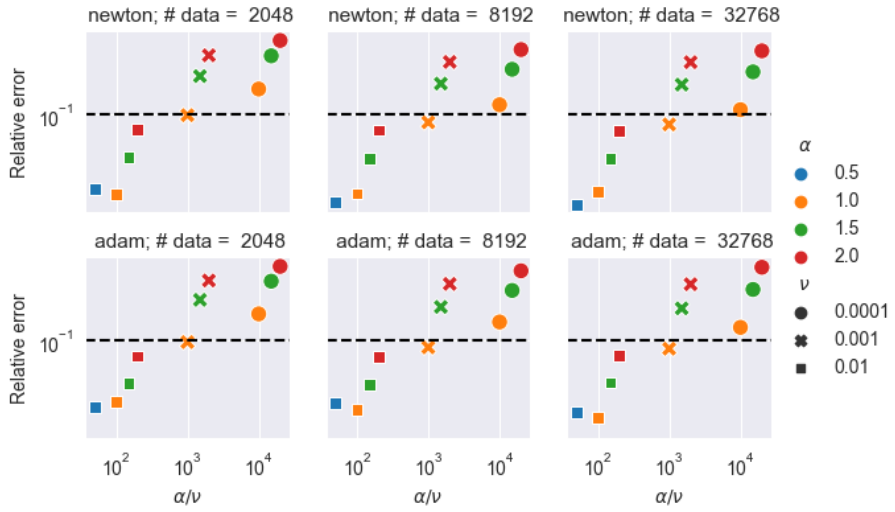

Figure 5. For varying amounts of training data (columns) and different PDE optimizers (rows), we plot the relative errors in the predicted solutions to the Burgers' equation.

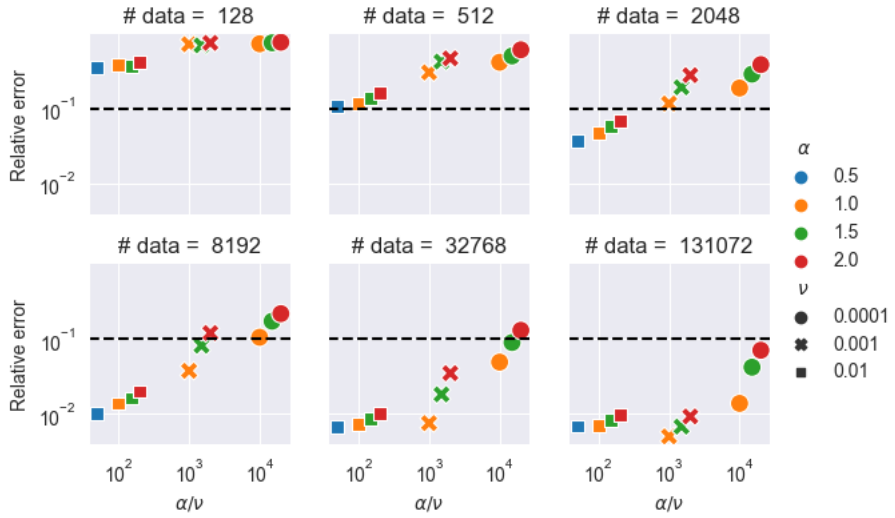

Figure 6. For varying amounts of training data, we plot the relative errors in estimating the Burgers' solution using the data-driven NN (Algorithm 1) strategy across different Burgers' parameters. The black dashed line indicates 10% or less error, our threshold for success.

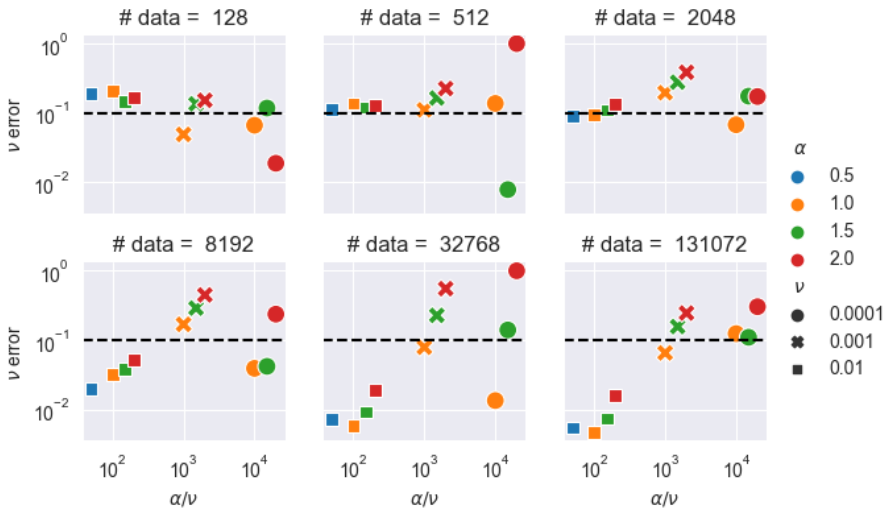

Figure 7. For varying amounts of training data, we plot the relative errors in estimating the convection coefficient α using the data-driven NN (Algorithm 1) strategy across different Burgers' parameters. The black dashed line indicates 10% or less error, our threshold for success.

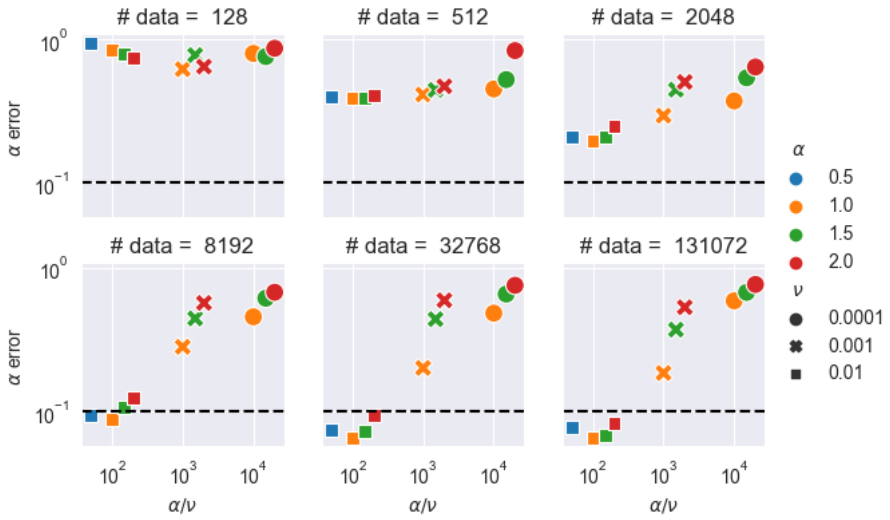

Figure 8. For varying amounts of training data, we plot the relative errors in estimating the diffusion coefficient ν, using the data-driven NN (Algorithm 1) strategy across different Burgers' parameters. The black dashed line indicates 10% or less error, our threshold for success.