Temporal Consistency Loss for Physics-Informed Neural Networks

Sukirt Thakur,[1] Maziar Raissi,[2] Harsa Mitra,[1] and Arezoo M. Ardekani*[1]
[1)]*School of Mechanical Engineering, Purdue University, West Lafayette, 47907, Indiana, USA*
[2)]*Department of Mathematics,University of California, Riverside, 92521, California, USA*

(*Electronic mail: Corresponding author: ardekani@purdue.edu.)

Physics-informed neural networks (PINNs) have been widely used to solve partial differential equations (PDEs) in a forward and inverse manner using neural networks. However, balancing individual loss terms can be challenging, mainly when training these networks for stiff PDEs and scenarios requiring enforcement of numerous constraints. Even though statistical methods can be applied to assign relative weights to the regression loss for data, assigning relative weights to equation-based loss terms remains a formidable task. This paper proposes a method for assigning relative weights to the mean squared loss terms in the objective function used to train PINNs. Due to the presence of temporal gradients in the governing equation, the physics-informed loss can be recast using numerical integration through backward Euler discretization. The physics-uninformed and physics-informed networks should yield identical predictions when assessed at corresponding spatio-temporal positions. We refer to this consistency as 'temporal consistency.' This approach introduces a unique method for training Physics-Informed Neural Networks (PINNs), redefining the loss function to allow for assigning relative weights with statistical properties of the observed data. In this work, we consider the two and three-dimensional Navier-Stokes equations and determine the kinematic viscosity using the spatio-temporal data on the velocity and pressure fields. We consider numerical datasets to test our method. We test the sensitivity of our method to the time step size, the number of timesteps, noise in the data, and spatial resolution. Finally, we use the velocity field obtained using Particle Image Velocimetry (PIV) experiments to generate a reference pressure field and test our framework using the velocity and pressure fields.

I. INTRODUCTION

Physics-informed neural networks[1,2] (PINNs) have become a popular method for solving a wide range of forward and inverse problems. While traditional deep learning methods are data intensive and do not consider the physics of the problem, PINNs leverage the prior information that we have in the form of governing partial differential equations (PDEs). Using the governing equations to regularize the optimization of parameters in PINNs allows us to train large networks with small datasets. This proves handy for problems in biological and engineering systems, as collecting data can be tedious and expensive.

Potential enhancements to Physics-Informed Neural Networks (PINNs) span five key dimensions: 1. Incorporating more complex physical phenomena and governing equations, 2. Tackling problems with intricate geometries, 3. Developing more effective loss functions, 4. Exploring novel neural network architectures tailored to PINN applications, and 5. Improving training techniques and optimization strategies for enhanced convergence and stability. While PINNs have been used to solve a whole range of multiphysics problems[3–5], there has been much interest in deploying PINNs to tackle problems in fluid mechanics[6–9]. PINN-based frameworks have been used to model high-speed aerodynamic flows[10], porous media flows[11], and biomedical flows[12]. Recently, PINNs have been used to solve non-Newtonian and complex fluid systems[13,14].

While vanilla feed-forward neural networks remain the most popular architecture, PINNs have been extended to use multiple feed-forward networks[15,16], convolution neural networks[17,18], recurrent neural networks[19,20], and Bayesian neural networks[21]. However, there are challenges associated with training PINNs. It is not straightforward to train PINNs with "stiff" PDEs and multiscale problems. There have been numerous efforts to tackle the problem of assigning relative weights to the different objectives. Apart from assigning relative weights through trial and error, the methods include learning rate annealing[22], minmax weighting[23], using the eigenvalues of the neural tangent kernel matrix[24] and using the soft self-attention mechanism[25]. In this work, we focus on a novel loss function and training process for PINNs by assigning relative weights to the loss terms using statistical properties of the observed data. Standardization is a widely adopted scaling technique in machine learning. Typically, standardization involves subtracting the mean and dividing by the standard deviation. However, for regression tasks, it is common to only divide by the standard deviation and hence, the relative weight of the loss term is one over the square of the standard deviation. However, for regression tasks, it is common to only divide by the standard deviation; hence, the loss term's relative weight is one over the square of the standard deviation. Thus, the relative weight of the loss term is one over the square of the standard deviation. Thus, relative weights for the loss term can be assigned for loss terms for regression. For PINNs, the goal is to achieve residual equations that ideally equal zero. In other words, the 'physics-informed' loss is a regression loss for a spatiotemporally uniform zero-value. However, Assigning a relative weight to the residual loss using standard deviation is not feasible when the target value is uniformly zero, as the standard deviation is zero. To address this challenge, our proposed method incorporates a consistency loss, resulting in a target variable with a non-zero standard deviation that is known a priori. This information enables more effective scaling of the regression of the data with the physics-informed term in the loss, thereby facilitating more effective training of the network parameters. Due to the presence of tempo-

ral gradients in the governing equation, the physics-informed loss can be recast using numerical integration through backward Euler discretization. Other discrete schemes can also be used; we focus on backward Euler discretization in this work without any loss of generality. This allows us to redefine the loss function and assign relative weights with one over the square of the standard deviation of the observed data. In this work, we consider the two-dimensional and three-dimensional Navier-Stokes equations which govern fluid flows. We obtain the viscosity using the velocity and pressure fields as the observations. We test the sensitivity and robustness of our method to the time step size, the number of timesteps, noise in the data, and spatial resolution for a numerical dataset in section III A. Here, the number of timesteps refers to the number of discrete time slices we randomly sample from. As our method works robustly, we benchmark our method against the experimental dataset of Particle Image Velocimetry (PIV) observations in section III B. Finally, we provide some concluding remarks and discuss the future scope of our work in section IV.

II. METHODOLOGY

A. Fluid Governing Equations

The conservation of mass for an incompressible fluid is given by

$$\nabla \cdot \boldsymbol{u} = 0, \tag{1}$$

where $\boldsymbol{u}$ is the fluid velocity vector. The conservation of momentum of an incompressible Newtonian fluid under isothermal, single-phase, transient conditions in the absence of a body force is given by

$$\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}\right) = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u}, \tag{2}$$

where ρ is the density of the fluid, $\boldsymbol{u}$ is the velocity vector, t is the time, p is the pressure, and ν is the kinematic viscosity. In this work, we consider kinematic pressure and drop the density term for the rest of the discussions. The vector form of the momentum equation in two dimensions in x and y directions is, respectively, given by

$$\begin{aligned} u_t + uu_x + vu_y &= -p_x + \nu(u_{xx} + u_{yy}), \\ v_t + uv_x + vv_y &= -p_y + \nu(v_{xx} + v_{yy}), \end{aligned} \tag{3}$$

where the subscripts denote the derivatives. The momentum equation in vector form in the x, y and z directions is, respectively, given by

$$\begin{aligned} u_t + uu_x + vu_y + wu_z &= -p_x + \nu(u_{xx} + u_{yy} + u_{zz}), \\ v_t + uv_x + vv_y + wv_z &= -p_y + \nu(v_{xx} + v_{yy} + v_{zz}), \\ w_t + uw_x + vw_y + ww_z &= -p_z + \nu(w_{xx} + w_{yy} + w_{zz}). \end{aligned} \tag{4}$$

We can decompose the momentum equation into two parts for each direction: the temporal derivative and the remaining terms. Let's define $\boldsymbol{f} = (f^x, f^y, f^z)$, where

$$\begin{aligned} f^x(\boldsymbol{u}, p; \nu) &= uu_x + vu_y + wu_z + p_x - \nu(u_{xx} + u_{yy} + u_{zz}), \\ f^y(\boldsymbol{u}, p; \nu) &= uv_x + vv_y + wv_z + p_y - \nu(v_{xx} + v_{yy} + v_{zz}), \\ f^z(\boldsymbol{u}, p; \nu) &= uw_x + vw_y + ww_z + p_z - \nu(w_{xx} + w_{yy} + w_{zz}). \end{aligned} \tag{5}$$

The temporal derivative can be computed using automatic differentiation or other numerical techniques. In this study, our focus is on employing backward Euler discretization, although alternative numerical methods could also be utilized. The backward Euler method approximates the temporal derivative (a') as

$$a'(t) \approx \frac{a(t) - a(t-h)}{h}, \tag{6}$$

for a step size h. In the following section, we discuss the construction of a physics-informed network incorporating the backward Euler discretization method for the time derivative in the momentum equations.

B. Physics informed neural networks

In this study, we considered velocity and kinematic pressure as the observables and trained the model to learn the value of kinematic viscosity. The kinematic pressure, defined as pressure divided by density, and kinematic viscosity, defined as viscosity divided by density, are used in the governing equations. Similarly, the model could have been used to learn density if static pressure was observable. We define the spatial coordinates in two and three dimensions as $\boldsymbol{x} = (x, y)$ and $\boldsymbol{x} = (x, y, z)$. We define the velocity field of an incompressible isothermal Newtonian fluid as

$$\boldsymbol{u}(t, \boldsymbol{x}) = (u(t, \boldsymbol{x}), v(t, \boldsymbol{x})), \tag{7}$$

in two dimensions and as

$$\boldsymbol{u}(t, \boldsymbol{x}) = (u(t, \boldsymbol{x}), v(t, \boldsymbol{x}), w(t, \boldsymbol{x})), \tag{8}$$

in three dimensions. Our observables at the N spatio-temporal data coordinates $\{(t_n, \boldsymbol{x}_n), n = 1, \ldots, N\}$ are the corresponding velocity and pressure fields. We define the velocity field in three dimensions as

$$\boldsymbol{u} = \nabla \times \boldsymbol{\psi}, \tag{9}$$

where $\boldsymbol{\psi}$ is a vector in three dimensions. We define the vector $\boldsymbol{\psi}$ with components ψ^1, ψ^2, and ψ^3. We get the velocity field as

$$\begin{aligned} u &= \psi^3_y - \psi^2_z \\ v &= \psi^1_z - \psi^3_x \\ w &= \psi^2_x - \psi^1_y, \end{aligned} \tag{10}$$

where u, v and w are the components of velocity in the x, y and z directions, respectively. By definition, the velocity field will

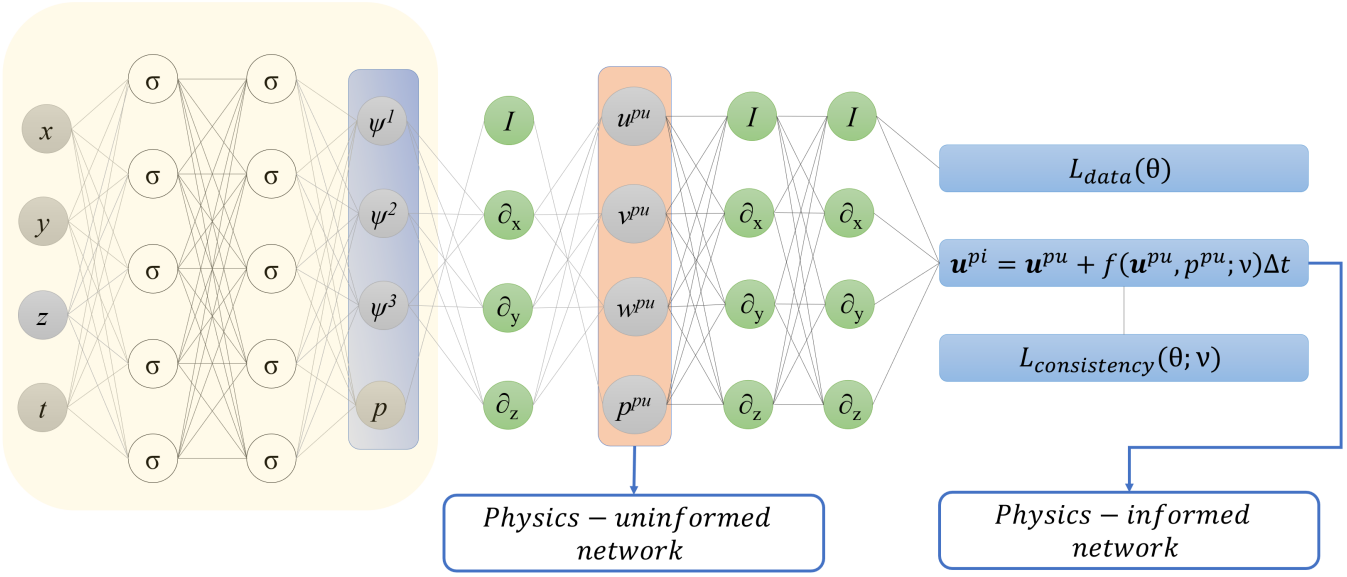

FIG. 1. The schematic of our framework to learn the viscosity from velocity and pressure fields in two dimensions. The network takes $t, \boldsymbol{x}$ as inputs and outputs the components of the vector field ψ and the pressure p. Here, I denotes the identity operator, and we compute the differential operators ∂x, ∂y, and ∂z using automatic differentiation. The physics-uninformed (denoted using the superscript 'pu') velocity field was then constructed using eq. (9). The physics-informed (denoted using the superscript 'pi') velocity field was constructed using the definition in eq. (13). The parameters of the neural network (θ) and the viscosity (ν) are then optimized to minimize the mean squared loss terms L_{data} and $L_{consistency}$.

then satisfy the continuity equation (1). In two dimensions, for the x and y components of velocity, we make the assumption that

$$u = \psi_y, v = -\psi_x, \tag{11}$$

for some latent function $\psi(t, \boldsymbol{x})$. We approximate the function $(t, x, y) \longmapsto (\psi, p)$ using a deep neural network with parameters θ for the two-dimensional case. Here p denotes the pressure field. For the three-dimensional case, a deep neural network with parameters θ was used to approximate the function $(t, x, y, z) \longmapsto (\psi^1, \psi^2, \psi^3, p)$. The divergence-free 'physics-uniformed' velocity field is constructed using (ψ^1, ψ^2, ψ^3) and eq. (9). The neural network architecture comprises 8 layers with 128 neurons in each layer for the 2D case, and 10 layers with 200 neurons in each layer for the 3D case. This configuration has been empirically determined to effectively solve 2D and 3D Navier-Stokes equations. We define the mean squared loss for regression over the velocity and pressure fields as

$$\begin{aligned} L_{data}(\theta) =& \mathbb{E}_{(t,\boldsymbol{x},\boldsymbol{u})}\left[\frac{|\boldsymbol{u}^{pu}(t,\boldsymbol{x};\theta) - \boldsymbol{u}|^2}{{\sigma_{\boldsymbol{u}}}^2}\right] + \\ & \mathbb{E}_{(t,\boldsymbol{x},p)}\left[\frac{|p^{pu}(t,\boldsymbol{x},;\theta) - p|^2}{{\sigma_p}^2}\right], \end{aligned} \tag{12}$$

where $\sigma_{\boldsymbol{u}}$ and σ_p are the standard deviations of the reference velocity field and the reference pressure field, respectively and the subscript 'pu' denotes a physics-uninformed network. Here $\mathbb{E}$ denotes the expectation approximated by the population mean (i.e., mean of the observations $t_n, x_n, y_n, z_n, u_n, v_n, w_n, p_n, n = 1, \ldots, N.$). The standard deviation is used to scale the loss terms using standardization. We now create physics-informed neural networks using backward Euler discretization for the time derivative. Other discrete schemes can also be used; we focus on backward Euler discretization in this work without any loss of generality. Using the definition of $\boldsymbol{f}$ in eq. (5), we use numerical integration using the backward Euler discretization (eq. (6)) to get

$$\begin{aligned} \boldsymbol{u}^{pi}(t,\boldsymbol{x};\Delta t,\theta,\nu) = \boldsymbol{u}^{pu}(t+\Delta t,\boldsymbol{x};\theta) + \Delta t\, \boldsymbol{f}(\boldsymbol{u}^{pu}(t+\Delta t,\boldsymbol{x};\theta) \\ p^{pu}(t+\Delta t,\boldsymbol{x};\theta);\nu), \end{aligned} \tag{13}$$

here the superscript 'pi' denotes a physics-informed network. We define the output of the feedforward neural network as the physics-uninformed network, responsible for fitting the reference data. Without any regularization term, this network may overfit the dataset, particularly when the number of samples is limited. Using the derivatives computed through backpropagation, we establish the physics-informed network, which employs the same network parameters (θ) as the physics-uninformed network. This approach integrates physical principles into the network training process to enhance generalization and model accuracy. There is no significant difference in the computational cost compared to regular 'vanilla' PINN. Since the physics-informed and uninformed networks evaluate the velocities at the same point t, x, y, they need to be consistent. We enforce this using a consistency loss

$$L_{consistency}(\theta;\nu) = \mathbb{E}_{(t,\boldsymbol{x})}\left[\frac{|\boldsymbol{u}^{pi}(t,\boldsymbol{x};\Delta t,\theta,\nu) - \boldsymbol{u}^{pu}(t,\boldsymbol{x};\theta)|^2}{{\sigma_{\boldsymbol{u}}}^2}\right]. \tag{14}$$

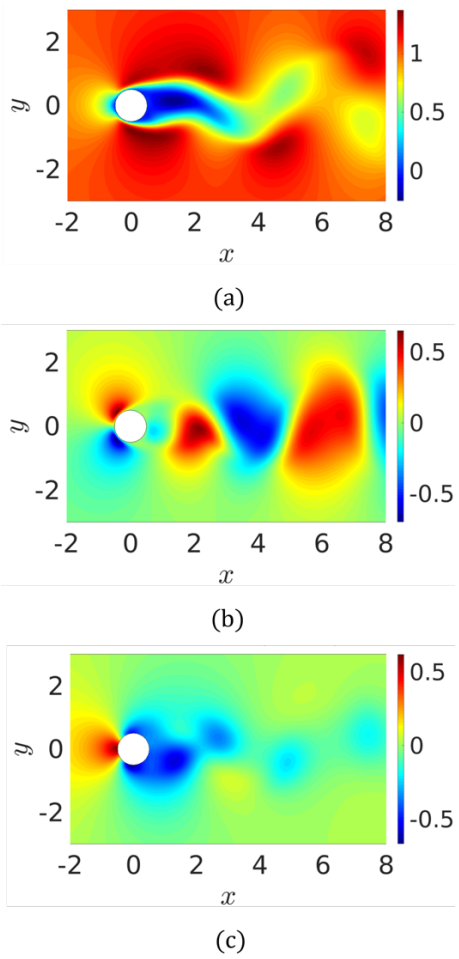

FIG. 2. A snapshot of the reference (a) x-velocity, (b) y-velocity and (c) pressure fields of the two-dimensional dataset.

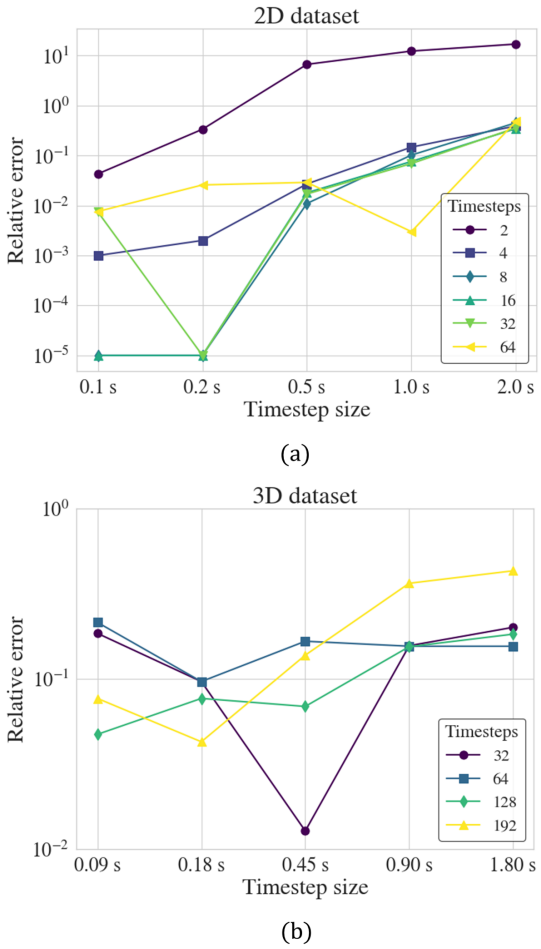

FIG. 3. Relative error for viscosity for different combinations of the number of timesteps and timestep size for the (a) two-dimensional and (b) three-dimensional numerical dataset.

We explain our framework using the schematic in Fig. 1. The 'physics-uninformed' and 'physics-informed' networks share the same parameters θ. The 'physics-informed' network is constructed using numerical integration of the momentum equation, using $\boldsymbol{f}$ as defined in eq. (5). The parameters θ and the viscosity ν is then optimized to minimize the following combined loss

$$L_{MSE}(\theta;\nu) = L_{data}(\theta) + L_{consistency}(\theta;\nu). \tag{15}$$

III. RESULTS

To test our method, we consider two and three-dimensional numerical datasets (section III A) and an experimental dataset (section III B). We generated the two-dimensional dataset using the open source CFD toolbox OpenFOAM[26] for the flow past a cylinder. A snapshot of the reference velocity and pressure fields of this dataset is shown in fig. 2. For the three-dimensional case, we look at the flow inside an aneurysm[27]. The three-dimensional dataset was generated using the spectral element method, and the dataset is available at https://github.com/maziarraissi/HFM. For the experimental dataset, we calculate the velocity field for water in a channel flow using PIVLab[28] by using seed particles. A PINN solver was then used to generate the pressure field using the known viscosity of water. We then used the velocity field from PIVlab and the pressure field from the PINN solver to test the method discussed in this paper.

A. Numerical datasets

For all the two-dimensional datasets in this section, we present the scalar fields ψ and pressure using an eight-layer deep, fully connected neural network with 128 neurons per hidden layer. For the three-dimensional case, we present ψ_1, ψ_2, ψ_3 and pressure using a ten-layer deep neural network with 200 neurons per hidden layer. We use the swish activation function. The use of other architectures might improve the results. A cosine learning rate schedule[29] was used in all

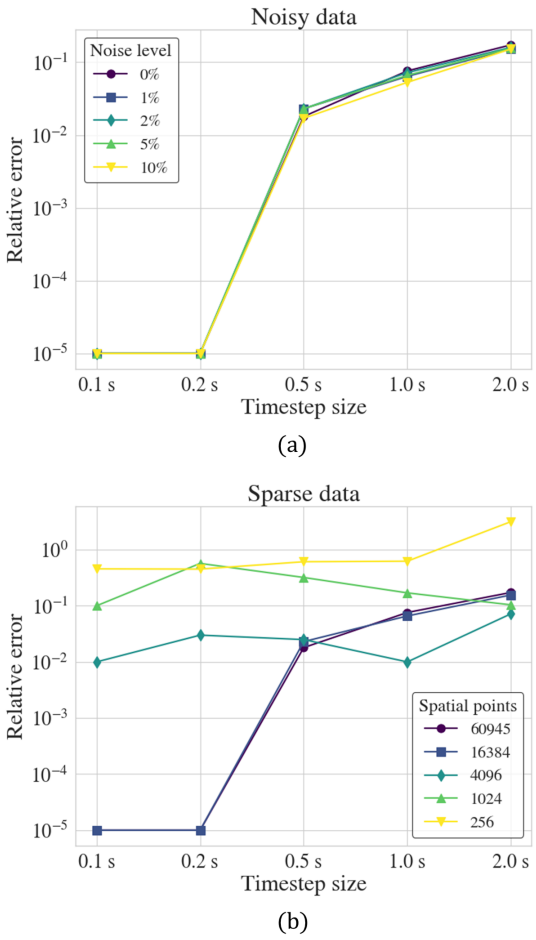

FIG. 4. Relative error for viscosity for (a) different combinations of the number of timesteps and noise and (b) different combinations of the number of timesteps and number of spatial points for the two-dimensional numerical dataset. We noticed that the addition of Gaussian noise does not have a significant effect on results. Our framework eventually breaks down when only 256 points are randomly sampled.

the runs reported in this work. We used a value of 2.5e-03 for η_{max} and 2.5e-06 for η_{min} to get the learning rate η as defined in the following equation

$$\eta = \eta_{min} + 0.5(\eta_{max} - \eta_{min})\left(1 + \cos\left(\frac{T_{cur}}{T_{max}}\pi\right)\right), \quad (16)$$

where T_{cur} is the current time step and T_{max} is the total timestep. For the two-dimensional case, we choose a mini-batch size of 1024 for the spatio-temporal point cloud inside the domain. The Adam optimizer[30] was used to optimize the parameters of the neural network. We ran 100,000 iterations of the Adam optimizer for each two-dimensional case, and every ten iterations of the Adam optimizer took about 0.15 seconds. We used the same learning rate schedule and mini-batch size for the three-dimensional runs. For the three-dimensional cases, we optimized the parameters using 360,000 iterations of the Adam optimizer, where ten iterations took about 0.54 seconds.

We first tested the sensitivity of our method to the timestep size and the number of time steps. Here, the number of timesteps refers to the number of discrete time slices we randomly sample from. We do this to test the sensitivity of our method to temporal resolution and the amount of data. The reference value for the dimensionless kinematic viscosity was 0.01, while the reference dimensionless kinematic viscosity was 0.01018 the three-dimensional dataset. We show the plot for the relative errors for different combinations of timestep size and number of timesteps for the two-dimensional and three-dimensional cases in fig. 3. We define the relative error for viscosity as

$$\mathbb{L}_{relative}(\nu_{true}, \nu_{pred}) = \left|\frac{\nu_{true} - \nu_{pred}}{\nu_{true}}\right|, \quad (17)$$

where ν_{true} and ν_{pred} are the true and predicted values for the viscosity, respectively. The numerical dataset for flow past a cylinder consists of 200 timesteps, with 60,945 spatial points at each timestep. For the aneurysm dataset, there are 300 timesteps, with 98,786 spatial points at each timestep. All spatial points were utilized in our sensitivity analysis involving varying timestep sizes and the number of timesteps. While the trend is not strictly monotonic, smaller timestep sizes generally lead to better results. Initially, increasing the number of timesteps improves performance; however, beyond a certain point, further increases in the number of timesteps do not yield additional improvements for both 2D and 3D cases. Our framework reports a low relative error for a wide range of combinations, as a relative error of less than 10^{-2} means that the predicted value for viscosity was between 0.0099 and 0.0101 for a true value of 0.01. In fig. 5, we compare the convergence between 'vanilla' PINN and PINN with temporal consistency using identical network shapes and sizes for the two-dimensional and three-dimensional cases. Although both networks approach the true value of 0.01 for the dimensionless kinematic viscosity for the 2D case, the PINN with temporal consistency exhibits slightly superior convergence. The better convergence rate of the temporal consistency PINN is demonstrated with the more challenging problem of the three-dimensional Navier-Stokes equations. For the three-dimensional case, not only does the temporal consistency PINN converge faster, but it also converges more accurately. We also analyze the physics-informed loss terms for the 'vanilla' PINN and the temporal consistency PINN. It is observed that the temporal consistency loss term exhibits smoother descent and reaches a lower optimum compared to the 'vanilla' PINN. While for the 2D case, both vanilla PINN and temporal consistency PINN converge to similar values, for the 3D case, the physics-informed loss term converges much better for temporal consistency PINN. To test the sensitivity of our method to noise, we added Gaussian noise to the two-dimensional dataset. We visualize the relative errors for viscosity using data from 16 timesteps with varying time step sizes of $0.1s, 0.2s, 0.5s, 1.0s$, and $2.0s$, across different levels of Gaussian noise, in fig. 4. We observed that the amount of Gaussian noise did not significantly affect the error, and our method worked well even when 10% Gaussian noise was added to the dataset. This result was in agreement with what

was observed for PINNs earlier[14]. This method proves highly effective when there is available data on the physics-informed variable under consideration, making it particularly suitable for many inverse problems. However, in scenarios where we are solving a forward problem with known initial and boundary conditions only, challenges may arise. For instance, if the initial condition is uniformly zero, extrapolating the standard deviation solely from this information might not accurately represent the standard deviation across the entire domain. These situations can pose difficulties for the proposed method.

The low sensitivity to Gaussian noise might result from many spatial points in the dataset. We trained our model on fewer spatial points to test this. We randomly sampled 60945, 16384, 4096, 1024, and 256 spatial data points at 16 time steps and added 5% Gaussian noise. We report the relative errors in fig. 4. We did not observe any discernible change in the performance of our framework when we reduced the resolution of spatial points from 60,945 to 16,384. The relative errors began to increase when we reduced the number of sampled data points to 4,096 per timestep. However, our approach continued to perform effectively with smaller timestep sizes. Significant degradation in performance was observed when we further reduced the random sampling to just 256 spatial points per timestep. Since our setup worked for sparse and noisy data, we next considered a real-world dataset obtained through experiments.

B. Experimental Dataset

Water seeded with 1.04 μm mean diameter and 1% solids fluorescent polystyrene (PS) beads (Bangs Laboratories Inc., IN, USA) at 2% (w/w) concentration is used for the experimental validation. As shown in Fig. 6, a syringe pump drives the fluid flow within an oblique channel of 1 mm width and 0.4 mm height (μ-Slide III^{3in1}, ibidi GmbH, Grafelfing, Germany). We applied water flow at 40 μl/min. A stable and laminar flow, with velocity along the flow direction being $\sim 0.0017\ m/s$ and Reynold's number of ~ 0.95 is developed. Stokes number for the flow was $\ll 0.1$, ensuring PS beads follow fluid streamlines closely maintaining a tracing accuracy error below 1%. A 520 nm laser using an inverted microscope coupled with a confocal system (Nikon, NY, USA) for imaging and used an oil immersion 60x (0.1083 μm/px) lens. We collected 3000 images at 5 ms intervals (200 fps). Experiments were performed at room temperature of 21°C.

For post-processing PIVlab MATLAB GUI is used[28]. We imported the images in the pairwise sequencing scheme. Image pre-processing using the PIVlab interface is also applied to remove the background light intensity. The 2-D velocity field is extracted in the x-y plane using the Fast Fourier Transform (FFT) window deformation algorithm, along with three passes, i.e., 128, 64, and 32 pixel interrogation areas. Finally, the mean- x and y velocity components are calculated and exported separately.

We use the velocity field obtained from PIVLab to generate a reference pressure field. Our framework then uses the veloc-

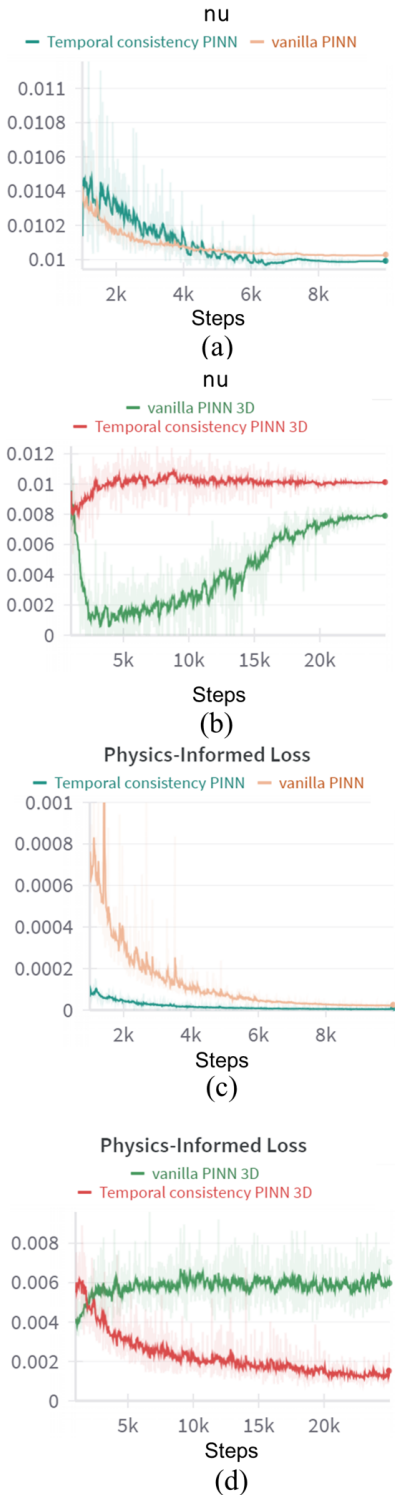

FIG. 5. Comparison of convergence of viscosity for the (a) two-dimensional and (b) three-dimensional Navier-Stokes equations between vanilla PINN and PINN with temporal consistency. While for the 2D case, the vanilla PINN converges to a very similar value for viscosity, for the 3D case, the prediction of the temporal consistency PINN is more accurate to the ground truth value of 0.01018. The better convergence of the temporal consistency PINN can be further demonstrated using the physics-informed loss term for the (c) 2D and (d) 3D cases. While the temporal consistency loss term converges faster for both cases, for the 3D case, it does significantly better than vanilla PINN.

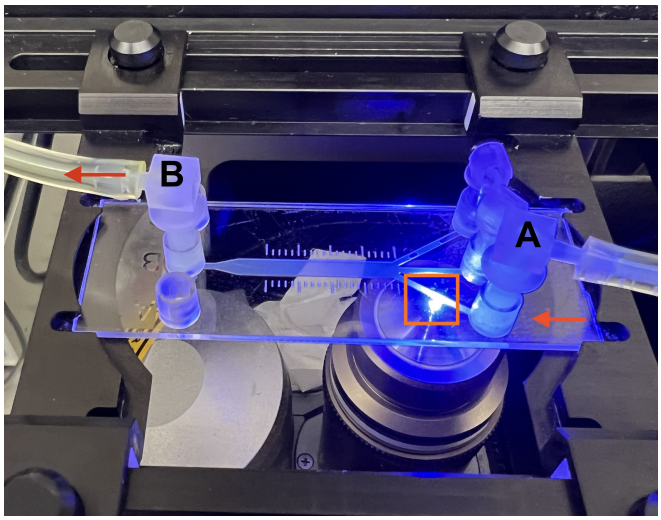

FIG. 6. The experimental setup with the μ-Slide III 3in1 (ibidi Inc., WI, USA) is used for the PIV measurements. The flow inlet and outlet are labeled as A and B, respectively. The interrogation area (not to scale) is also represented using the orange square. During the experiment, the other two inlets were closed using ibidi luer locks.

ity and pressure fields to predict water viscosity at room temperature. We used an eight-layer deep, fully connected neural network with 128 neurons per hidden layer. We used the learning schedule described in section III A, and the parameters of the network were optimized using 800,000 iterations of the Adam optimizer. The timestep size for the experimental dataset was 0.005 seconds. The reference value for the water viscosity at room temperature is 0.01 poise[31], and the value we get from our model is 0.00977 poise.

Studying the neural tangent kernel of PINNs has demonstrated that PINNs have an implicit bias to minimize the PDE residuals at later times before fitting the initial data[32]. Our method ensures that the physics-informed loss depends on the data at the timestep before, which adds causality to the training method. We highlight this result for its novelty in two key aspects. Firstly, this study represents one of the few instances where PINNs have been successfully applied to real-world data. Reporting this outcome is crucial for building confidence in the method's practical application. Secondly, our framework offers a valuable auxiliary tool for experimentalists seeking to extract additional insights from experimentally acquired velocity fields. A potential area for future exploration involves extending this framework to learn additional properties such as pressure and density from observations on the velocity. While our framework does not provide all the answers, it represents a significant step towards harnessing the potential of PINNs to extract richer information from experimental data.

IV. CONCLUSIONS AND FUTURE WORK

It is generally challenging to assign relative weights to the loss terms while training physics-informed neural networks We propose a novel solution for this challenge. By defining the physics-informed network through numerical integration using backward Euler discretization, we can use the data's statistical properties to get the loss terms' relative weights. In this work, we consider the two and three-dimensional Navier-Stokes equations and determine the kinematic viscosity using spatio-temporal data on the velocity and pressure fields.

For the two-dimensional case, we look at the flow past a cylinder and flow in an aneurysm for the three-dimensional case. We test the sensitivity and robustness of our method against the timestep size, the number of timesteps, noise in the data, and the spatial data resolution. We compared the convergence of the temporal consistency PINN with 'vanilla' PINN, and demonstrated faster and more accurate convergence of the temporal consistency PINN for the more challenging three-dimensional Navier-Stokes equations. Since our method worked well for a wide range of numerical data, we tested our method using experimental data.

We used the velocity field from experimental PIV measurements of a channel flow to generate a reference pressure field. We tested our framework using this velocity and reference pressure fields to get water viscosity at room temperature. We demonstrated that our framework worked well with an experimental dataset. This work uses spatio-temporal data on the pressure and velocity fields as input. We believe this approach shows great promise for 'stiff' Partial Differential Equations (PDEs) and scenarios requiring enforcement of numerous constraints. These are challenges that traditional 'vanilla' PINNs may struggle to overcome. The ability to standardize loss terms is crucial for effectively training neural networks. This standardization process ensures balanced learning across diverse loss scales, optimizing the network's training performance. This method could potentially benefit from the implementation of more advanced numerical techniques such as explicit and implicit Runge-Kutta integration methods. We encourage readers to explore these possibilities as part of future work. Additionally, for future investigations, exploring the use of only the velocity field as input to solve for the pressure field could be pursued. This approach would involve directly utilizing velocity field data from Particle Image Velocimetry (PIV) measurements to learn viscosity and pressure fields for both two-dimensional and three-dimensional flows.

V. ACKNOWLEDGEMENTS

A.M.A. acknowledges financial support from the National Science Foundation (NSF) through Grant No. CBET-2141404.

[1]M. Raissi, P. Perdikaris, G. 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 (2019) 686–707. doi:10.1016/j.jcp.2018.10.045. URL https://doi.org/10.1016/j.jcp.2018.10.045

[2]M. Raissi, Deep hidden physics models: Deep learning of nonlinear partial differential equations, Journal of Machine Learning Research 19 (2018) 1–24.

[3]S. Cai, Z. Wang, S. Wang, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks for heat transfer problems, Journal of Heat Transfer 143 (6) (2021) 1–15. doi:10.1115/1.4050542.

[4]T. Kadeethum, T. M. Jørgensen, H. M. Nick, Physics-informed neural networks for solving nonlinear diffusivity and Biot's equations, PLoS ONE 15 (5) (2020) 1–28. doi:10.1371/journal.pone.0232683.

[5]O. Hennigh, S. Narasimhan, M. A. Nabian, A. Subramaniam, K. Tangsali, Z. Fang, M. Rietmann, W. Byeon, S. Choudhry, NVIDIA SimNet™: An AI-Accelerated Multi-Physics Simulation Framework, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 12746 LNCS (2021) 447–461. doi:10.1007/978-3-030-77977-1{_}36.

[6]X. Jin, S. Cai, H. Li, G. E. Karniadakis, NSFnets (Navier-Stokes Flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations (Hui Li).

[7]C. J. Arthurs, A. P. King, Active training of physics-informed neural networks to aggregate and interpolate parametric solutions to the Navier-Stokes equations, Journal of Computational Physics 438 (2021) 110364. doi:10.1016/j.jcp.2021.110364.
URL https://doi.org/10.1016/j.jcp.2021.110364

[8]S. Cuomo, V. Schiano, D. Cola, G. Rozza, M. Raissi, Scientific Machine Learning through Physics-Informed Neural Networks : Where we are and What ' s next.

[9]S. Cai, Z. Mao, Z. Wang, M. Yin, G. E. Karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: a review, Acta Mechanica Sinica/Lixue Xuebao 37 (12) (2021) 1727–1738. doi:10.1007/s10409-021-01148-1.
URL https://doi.org/10.1007/s10409-021-01148-1

[10]Z. Mao, A. D. Jagtap, G. E. Karniadakis, Physics-informed neural networks for high-speed flows, Computer Methods in Applied Mechanics and Engineering 360 (2020) 112789. doi:10.1016/j.cma.2019.112789.
URL https://doi.org/10.1016/j.cma.2019.112789

[11]M. M. Almajid, M. O. Abu-Al-Saud, Prediction of porous media fluid flow using physics informed neural networks, Journal of Petroleum Science and Engineering 208 (PA) (2022) 109205. doi:10.1016/j.petrol.2021.109205.
URL https://doi.org/10.1016/j.petrol.2021.109205

[12]G. Kissas, Y. Yang, E. Hwuang, W. R. Witschey, J. A. Detre, P. Perdikaris, Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks, Computer Methods in Applied Mechanics and Engineering 358 (2020) 112623. doi:10.1016/j.cma.2019.112623.
URL https://doi.org/10.1016/j.cma.2019.112623

[13]M. Mahmoudabadbozchelou, G. E. Karniadakis, S. Jamali, nn-PINNs: Non-Newtonian physics-informed neural networks for complex fluid modeling, Soft Matter 18 (1) (2022) 172–185. doi:10.1039/d1sm01298c.

[14]S. Thakur, M. Raissi, A. M. Ardekani, ViscoelasticNet: A physics informed neural network framework for stress discovery and model selection, Journal of Non-Newtonian Fluid Mechanics 330 (June) (2024) 105265. doi:10.1016/j.jnnfm.2024.105265.
URL https://doi.org/10.1016/j.jnnfm.2024.105265

[15]E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics, Computer Methods in Applied Mechanics and Engineering 379 (2021) 113741. doi:10.1016/j.cma.2021.113741.
URL https://doi.org/10.1016/j.cma.2021.113741

[16]B. Moseley, A. Markham, T. Nissen-Meyer, Finite Basis Physics-Informed Neural Networks (FBPINNs): a scalable domain decomposition approach for solving differential equations (2021).
URL http://arxiv.org/abs/2107.07871

[17]H. Gao, L. Sun, J. X. Wang, PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain, Journal of Computational Physics 428 (2021) 110079. doi:10.1016/j.jcp.2020.110079.
URL https://doi.org/10.1016/j.jcp.2020.110079

[18]Z. Fang, A High-Efficient Hybrid Physics-Informed Neural Networks Based on Convolutional Neural Network, IEEE Transactions on Neural Networks and Learning Systems (2021) 1–13doi:10.1109/TNNLS.2021.3070878.

[19]R. Zhang, Y. Liu, H. Sun, Physics-informed multi-LSTM networks for metamodeling of nonlinear structures, Computer Methods in Applied Mechanics and Engineering 369 (2020) 113226. doi:10.1016/j.cma.2020.113226.
URL https://doi.org/10.1016/j.cma.2020.113226

[20]Y. A. Yucesan, F. A. Viana, Hybrid physics-informed neural networks for main bearing fatigue prognosis with visual grease inspection, Computers in Industry 125 (2021) 103386. doi:10.1016/j.compind.2020.103386.
URL https://doi.org/10.1016/j.compind.2020.103386

[21]L. Yang, X. Meng, G. E. Karniadakis, B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data, Journal of Computational Physics 425 (2021) 109913. doi:10.1016/j.jcp.2020.109913.
URL https://doi.org/10.1016/j.jcp.2020.109913

[22]S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient pathologies in physics-informed neural networks (2020) 1–28.
URL http://arxiv.org/abs/2001.04536

[23]D. Liu, Y. Wang, A Dual-Dimer method for training physics-constrained neural networks with minimax architecture, Neural Networks 136 (2021) 112–125. doi:10.1016/j.neunet.2020.12.028.
URL https://doi.org/10.1016/j.neunet.2020.12.028

[24]S. Wang, X. Yu, P. Perdikaris, When and why PINNs fail to train: A neural tangent kernel perspective, Journal of Computational Physics 449 (2022) 1–29. doi:10.1016/j.jcp.2021.110768.

[25]L. McClenny, U. Braga-Neto, Self-adaptive physics-informed neural networks using a soft attention mechanism, CEUR Workshop Proceedings 2964 (2021).

[26]H. G. Weller, G. Tabor, H. Jasak, C. Fureby, A tensorial approach to computational continuum mechanics using object-oriented techniques, Computers in Physics 12 (6) (1998) 620. doi:10.1063/1.168744.

[27]M. Raissi, A. Yazdani, G. E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations (C) (2020) 1–5. doi:10.1126/science.aaw4741.
URL https://science.sciencemag.org/content/367/6481/1026/tab-pdf

[28]W. Thielicke, E. J. Stamhuis, PIVlab – Towards User-friendly, Affordable and Accurate Digital Particle Image Velocimetry in MATLAB, Journal of Open Research Software 2 (2014). doi:10.5334/jors.bl.

[29]I. Loshchilov, F. Hutter, SGDR: Stochastic gradient descent with warm restarts, 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings (2017) 1–16.

[30]D. P. Kingma, J. L. Ba, Adam: A method for stochastic optimization, 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings (2015) 1–15.

[31]I. F. Swindells, J. R. Coe, T. B. Godfrey, Absolute Viscosity of Water at 20 ° C 48 (1) (1952).

[32]S. Wang, S. Sankaran, P. Perdikaris, Respecting causality for training physics-informed neural networks, Computer Methods in Applied Mechanics and Engineering 421 (February) (2024). doi:10.1016/j.cma.2024.116813.