Proceedings of the 1st International Symposium on AI and Fluid Mechanics
Paper No S2 P6

Differentiable Simulation for Inverse-like Fluid Flow Problems

Stefan Posch[1], Marian Staggl[2], Wolfgang Sanz[2]

[1] *Large Engines Competence Center, Graz, Austria*
[2] *Institute of Thermal Turbomachinery and Machine Dynamics, Graz University of Technology, Austria*
Corresponding author: Stefan Posch (stefan.posch@lec.tugraz.at)

Keywords: Differentiable simulation, neural network, Burgers equation, Kovasznay flow, lid-driven cavity

Abstract. Inverse problems aim to estimate hidden parameters of a mathematical model using observed data. Deep learning models, particularly physics-informed neural networks (PINNs), have shown great potential for solving such problems. PINNs integrate physical laws into the loss function, ensuring solutions match observed data while adhering to governing physical principles, enhancing accuracy and reliability. Typically, PINNs predict unavailable variables through physics-informed loss functions. This work explores a different scenario: approximating the function $\underline{m} = g(\underline{k})$, where $\underline{k}$ are characteristic parameters to describe $\underline{m}$. However, $\underline{m}$ is not explicitly represented in the available data. Instead, a known physical relation $\underline{u} = f(\underline{x}, t, \underline{m})$, expressed by a partial differential equation (PDE), and data for $\underline{u}$ are available. To train the network, a custom loss function incorporates the PDE solution via a differentiable numerical solver, enabling gradient computation throughout the PDE solution process for effective training. This method has been applied to three flow problems, namely the Burgers equation, the Kovasznay flow and the lid-driven cavity flow, demonstrating accurate parameter identification and robustness. Potential applications include learning material laws and virtual sensing, where indirect measurements infer critical system parameters.

1. Introduction

In several areas of science and engineering such as fluid mechanics, the accurate recovery of hidden multi-dimensional model parameters from indirect observations are a common issue [1]. These so-called inverse problems can be formalized as solving an equation in the form

$$\underline{u} = f(\underline{m}), \tag{1}$$

where $\underline{u} \in \underline{Y}$ are the observed data and $\underline{m} \in \underline{X}$ are the model parameters to determine. The mapping $f : \underline{X} \quad \underline{Y}$, is the forward operator [1]. To solve this inverse problem, several strategies can be employed. Traditional techniques such as regularization methods solve the forward problem multiple times in order to get the required information for the inverse problem. Lin et al. [2] estimated the viscosity of a power-law fluid combining temperature and pressure measurements, and a numerical model of the investigated system. The numerical model describes the viscous heating and heat convection used for the inverse analysis of the viscosity. A further regularization-based application for solving an inverse fluid flow problem was shown by Ouaissa et al. [3]. The authors presented a formulation for determining the fluid velocity and the flux over a part of the boundary by introducing given measurements on the remaining part of the inverse Cauchy problem governed by Stokes equation. Using the adjoint gradient method to solve the derived regularized solution, the proposed method is highly efficient. Kontogiannis et al. [4] solved the inverse Navier–Stokes problem for the joint velocity field reconstruction and boundary segmentation of noisy flow velocity images. In their work, the authors used a Bayesian framework with Gaussian random fields to regularize the problem and to estimate the uncertainties of the unknowns by a quasi-Newton method. Their outcomes indicate that the method successfully reconstructs and segments noisy images highly efficient.

Recent works have underlined the potential of using physics-informed neural networks (PINNs)[5] for inverse problems. Raissi et al. [5] already showed the use of PINNs for inverse problems in their initial work by approximating unknown parameters of the Navier-Stokes equations using noisy data. In one of their followed works, Raissi et al. [6] not only showed to estimate system parameters but also the possibility to predict the velocity and pressure field based on a given passive scalar. Jagtap et al. [7] applied the PINN framework for inverse problems in supersonic flows. In addition to the compressible Euler equations, the authors enforced the entropy conditions to obtain viscosity solutions. Furthermore, the authors reported that domain decomposition allows to use neural networks in each subdomain yielding convergence enhancement in case of high-gradient solutions as for shock problems. Bai et al. [8] gave an overview on strategies to improve the performance of PINNs for inverse problems. They combined adaptive activation function, loss function and sampling strategies. Tests on parameter identification in different partial differential equations (PDEs) such as Burgers equation and Navier-Stokes equation showed that their proposed strategy is able to reduce the relative error of the unknown parameters by up to two orders of magnitude compared to vanilla PINNs. Karnakov et al. [9] pointed out several drawbacks of using PINNs, in particular for inverse problems motivating for the development of a method based on optimizing a discrete loss (ODIL). The authors pointed out that PINNs often cannot match conventional numerical methods for well-posed forward PDE problems due to their high computational cost, which stems from evaluating a dense network at every collocation point and lacking the sparse structures typical of grid-based methods. They also do not incorporate physically motivated numerical techniques (e.g., upwind schemes) that can expedite convergence and ensure stability. Moreover, higher-order PDEs pose additional challenges because nested automatic differentiation grows exponentially with the order of derivatives, and many proposed architectural improvements do not guarantee a universal improvement in accuracy or training speed. ODIL minimizes a cost function for the discrete forms of PDEs using gradient-based and Newton's methods. The results of Karnakov et al. [9] showed that ODIL outperforms PINNs in terms of computational efficiency and accuracy on several inverse and ill-posed problems.

The present work addresses not only the estimation of model parameters, denoted as $\underline{m}$, from observed data but also the simultaneous learning of an operator function g that maps a given input parameter set $\underline{k}$ to $\underline{m}$ in a single step. The operator function g is modeled using a neural network, represented as

$$\underline{m} = g_\theta(\underline{k}), \tag{2}$$

where g_θ denotes the operator function depending on the neural network weights θ. The inverse problem is given by

$$\underline{u} = f(\underline{x}, t, \underline{m}), \tag{3}$$

where $\underline{u}$ is the observed data and f describes the underling physical system in form of a PDE. To enable the neural network training, the forward solution of f is integrated into the loss function using standard numerical schemes. Crucially, this approach requires the forward solver to be differentiable. Differentiable solvers facilitate the computation of gradients of the solution with respect to the problem parameters, enabling the use of common gradient-based optimization methods widely adopted in machine learning. For instance, List et al. [10] demonstrated the utility of differentiable fluid solvers in learning turbulence models from direct numerical simulation (DNS) data. In their approach, a convolutional neural network (CNN) predicts a corrective forcing term, and the integration with a differentiable solver allows the model to achieve excellent agreement with *a posteriori* DNS data validation. Further applications of differentiable flow solvers have been explored in [11, 12, 13].

The present study builds on this concept by combining a neural network to predict an unknown parameter $\underline{m}$ based on input features, while integrating a differentiable solver within the network loss function. This approach eliminates the need for direct training data for $\underline{m}$, as the network is trained by solving the inverse problem directly. The work is structured as follows: In Section 2, the methodology and the design of the ML framework is described. Section 3 contains the application of the methodology on different settings namely the Burgers equation, the Kovasznay flow and the lid driven cavity problem. In section 4, the results are discussed, and an outlook on further developments of the methodology is provided.

2. Methodology

2.1. Governing equations

To show the integration of neural networks with differentiable solvers, an inverse problem is formulated to estimate diffusion parameters, specifically the viscosity v, based on observable data. Two problem settings are considered: the viscous Burgers equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \frac{1}{\mathrm{Re}}\frac{\partial^2 u}{\partial x^2} \tag{4}$$

and the non-dimensional 2D incompressible Navier-Stokes equations:

$$\begin{aligned} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial v} &= -\frac{\partial p}{\partial x} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \\ \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial v} &= -\frac{\partial p}{\partial y} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \end{aligned} \tag{5}$$

where the Reynolds number Re is defined as:

$$\mathrm{Re} = \frac{UL}{v}. \tag{6}$$

2.2. Viscosity model

In order to demonstrate the ability of the method to learn material models, the well-known formulation of Sutherland describing the temperature dependence of the dynamic viscosity μ is used as a reference value. The formulation is given by

$$\mu = \mu_0 \left(\frac{T}{T_0}\right)^{3/2} \frac{T_0 + S}{T + S}, \tag{7}$$

where μ_0 is the dynamic viscosity at the reference temperature T_0 and S is an effective temperature known as the Sutherland constant. For the present study, a binary mixture of gases namely nitrogen (N_2) and oxygen (O_2) in a ratio of 80/20 is used. The mixture dynamic viscosity μ_{mix} is calculated based on the mixture rule of Wilke [14]:

$$\mu_{\mathrm{mix}} = \sum_{i=1}^{n} \frac{x_i \mu_i}{\phi_i} \quad \text{with} \quad \phi_i = \sum_{j=1} x_j \Phi_{ij} \tag{8}$$

and Φ_{ij} is given by:

$$\Phi_{ij} = \frac{\left[1 + \sqrt{\frac{\mu_i}{\mu_j}}\left(\frac{M_j}{M_i}\right)^{\frac{1}{4}}\right]^2}{\sqrt{8}\sqrt{1 + \frac{M_i}{M_j}}}, \tag{9}$$

where x_i, M_i and μ_i are the model fraction, molar mass and dynamic viscosity of component i. To calculate the kinematic viscosity v required to determine Re and consequently to solve the considered governing equations, the density of the mixture ρ_{mix} is calculated by the ideal gas law and the mixture molar mass according to:

$$\rho_{\text{mix}} = \frac{pM_{\text{mix}}}{RT} \quad \text{with} \quad M_{\text{mix}} = \sum_{i=1}^{n} x_i M_i, \tag{10}$$

where p is the system pressure specified with 1 bar and R is the universal gas constant. The data for the application examples in the present study are generated using a temperature range from 300K to 500K.

2.3. Model setup

The objective is to train a neural network N_θ to approximate the material model for ν as a function of input feature, in the present case the temperature T, expressed as $\nu = \mathsf{N}_\theta(T)$. In the considered problem settings, the observable data consists of u for the Burgers equation and the velocity components $[u, v]$ for the Navier-Stokes equations. To train the neural network, an appropriate loss function must be defined. This is achieved by solving the governing PDEs using standard numerical schemes, where the neural network output serves as a parameter for computing the field values. The loss function is formulated based on the mean absolute error (MAE) between the predicted and true values of u or $[u, v]$, depending on the application case. By ensuring that the numerical solver of the PDE is differentiable, the method enables the backpropagation of optimization gradients through multiple solver steps and neural network evaluations, facilitating efficient training of the network. This approach ensures that the learned parameter model is consistent with the underlying physics of the system. Fig. 1 shows a schematic illustration of the ML framework. The entire machine learning framework, including the numerical solver, is imple-

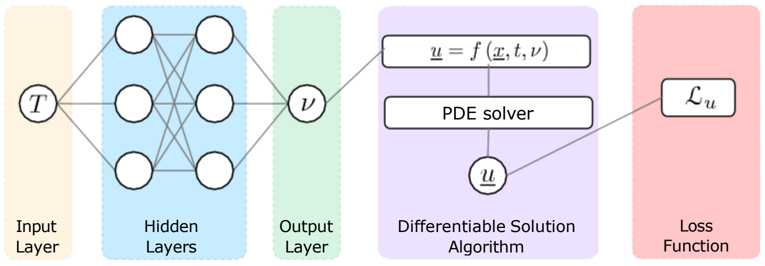


Figure 1: Illustration of the ML framework.

mented using PyTorch, which facilitates the use of automatic differentiation (AD) for efficient gradient computation. To reconstruct the viscosity model, a neural network with two hidden layers, each containing 24 neurons, is employed. The hidden layers utilize the *tanh* activation function, while the output layer applies the *sigmoid* activation function. The use of the *sigmoid* function in combination with an *a priori* scaling ensures that the predicted viscosity values remain within a bounded range, preventing the neural network from generating values unsuitable for the numerical solver used in the loss function (e.g., predicting turbulent flow while using a solver designed for laminar flow). However, this approach assumes prior knowledge of the expected viscosity range, requiring an appropriate scaling function for accurate predictions. The data is split into train and test set using a ratio of 80/20 throughout all considered examples. A learning rate scheduler using a factor of 0.999 in each epoch is utilized. The training process covers 300 epochs using the Adam optimizer. A more detailed description of the numerical schemes used to solve the governing PDEs is provided for each application example in the following section.

3. Applications

To demonstrate the performance of the methodology, three application examples are investigated namely the Burgers equation, the Kovasznay flow and the lid driven cavity flow. Details about the numerical schemes for designing the differentiable solver, the generation of the required data and the results obtained are shown in the following.

3.1. *Burgers equation*

Data generation. The Burgers equation is considered with the initial conditions

$$u(x,0) = \sin(\pi x), \qquad 0 < x < 1$$

and homogeneous boundary conditions

$$u(0,t) = u(1,t), \qquad t > 1.$$

The required data set for training and testing is generated by the analytical solution using the Cole-Hopf transformation [15, 16] given by

$$u(x,t) = 2\pi\nu \frac{\sum_{n=1}^{\infty} nA_n \sin(n\pi x)\exp\left(-\nu n^2\pi^2 t\right)}{A_0 + \sum_{n=1}^{\infty} nA_n \cos(n\pi x)\exp\left(-\nu n^2\pi^2 t\right)}, \tag{11a}$$

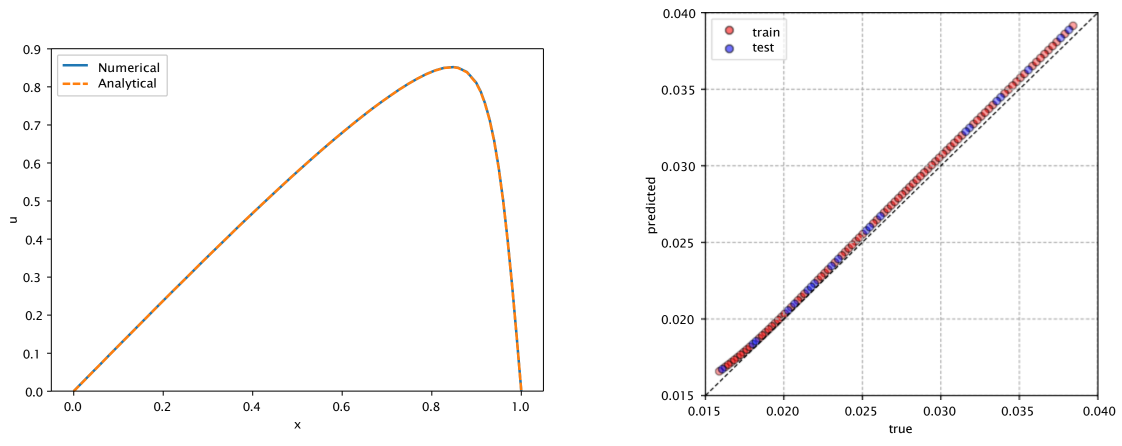
with the Fourier coefficients

$$A_n = \begin{cases} \int_0^1 \exp\left\{(2\pi\nu)^{-1}\left[\cos(\pi x) - 1\right]\right\} dx & n = 0 \\ \int_0^1 \exp\left\{(2\pi\nu)^{-1}\left[\cos(\pi x) - 1\right]\right\} \cos(n\pi x)\, dx & n \neq 0. \end{cases} \tag{11b}$$

To obtain values of Re to sustain stable results of the numerical solver, the characteristic velocity U in Eq. 6 is set to 10^{-3} ms^{-1}.

Numerical solver. The numerical solution of Eq. 4 follows standard discretization techniques commonly used for this type of equation. The diffusion term is approximated using a central difference scheme, while a first-order upwind scheme is employed for the convection term. For temporal discretization, the Crank-Nicolson scheme is applied, providing a balance between accuracy and stability. The computational domain is discretized into 50 spatial cells and 200 time steps, ensuring adequate resolution to capture the dynamics of the solution.

Results. To evaluate the accuracy of the differentiable numerical solver, the viscous Burgers' equation is solved for an arbitrary viscosity value, comparing the numerical solution with the analytical solution. Figure 2 (a) illustrates this comparison, demonstrating a high level of agreement between both solutions. The results confirm that the numerical solver accurately captures the underlying physics of the problem, ensuring a reliable basis for training the ML model. In Figure 2 (b), the performance of the ML model is assessed by comparing the true viscosity values with the predicted values obtained from the trained network. The results show that the predicted values align closely with the true values, showing a slight overprediction of the true values probably caused by deviations between the analytical and numerical solution. Nevertheless, the results indicate that the ML model effectively learns the underlying mapping between input features and viscosity, successfully solving the inverse problem.



(a) (b)

Figure 2: Results of the Burgers equation application case: (a) comparison between numerical solver and analytical solution for $\nu = 0.027$ m²/s, (b) predicted vs. true ν in m²/s.

3.2. Kovasznay flow

Data generation. The Kovasznay flow is a well-known analytical solution of the 2D incompressible Navier-Stokes equations for steady-state laminar flow behind a periodic array of vortices. It is particularly useful as a benchmark for validating CFD solvers. Kovasznay found an analytical solution for the velocity and pressure fields of this specific flow [17] given by

$$u(x, y) = 1 - \exp(\lambda x/L)\cos\left(\frac{2\pi y}{L}\right) \tag{12a}$$

$$v(x, y) = \frac{\lambda}{2\pi}\exp(\lambda x/L)\sin\left(\frac{2\pi y}{L}\right) \tag{12b}$$

$$p(x, y) = -\frac{1}{2}[1 - \exp(2\lambda x)] \tag{12c}$$

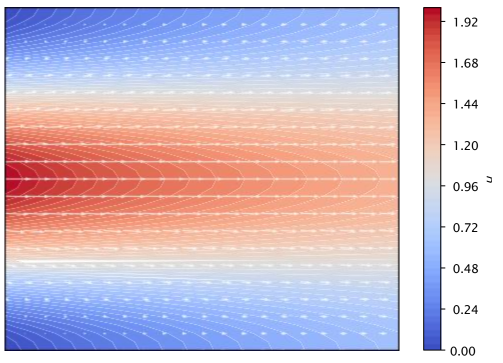
where

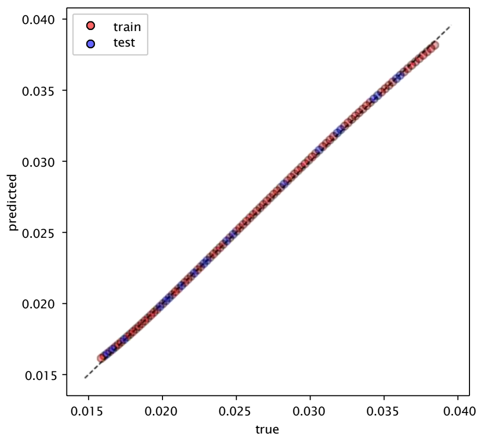
$$\lambda = \frac{1}{2}\left(\mathrm{Re} - \sqrt{\mathrm{Re}^2 + 16\pi^2}\right) \tag{12d}$$

is a decay parameter dependent on Re. The setting of the original paper where the distance between the vortices is used as the characteristic length L for non-dimensionalization. The characteristic velocity is set to 10^{-3} ms^{-1} to ensure that the Reynolds number remains within the range where the flow is laminar and the analytical solution is valid.

Numerical solver. The numerical solution of the Navier-Stokes equations is based on finite volume discretization. A first-order scheme is employed for the convection term to ensure numerical stability, while a second-order scheme is used for the diffusion term to improve accuracy. The pressure-velocity coupling required for solving the incompressible Navier-Stokes equations is handled through an iterative solver. The computational grid consists of 42 cells in both the x and y directions, providing a relatively coarse resolution. However, this resolution is sufficient to accurately capture the vortex structures, ensuring that the numerical solution remains consistent with the analytical Kovasznay flow solution, thus enabling an effective loss formulation that integrates both numerical and analytical data.

Results. Figure 3 (a) illustrates the velocity field computed by the numerical solver for an arbitrary viscosity value, demonstrating the solver's ability to accurately capture the flow characteristics. Comparing the numerical results with the analytical values leads to a mean relative L2 error of 0.002 for u and 0.011 for v, respectively. The ML-predicted viscosity values are compared against the true values as shown in Figure 3 (b). As the results indicate, a strong correlation between predicted and true viscosity values is obtained proving that the proposed method is able to extract the required features from the analytical solution although the numerical solver relies on a relativly rough computational resolution.





(a) (b)

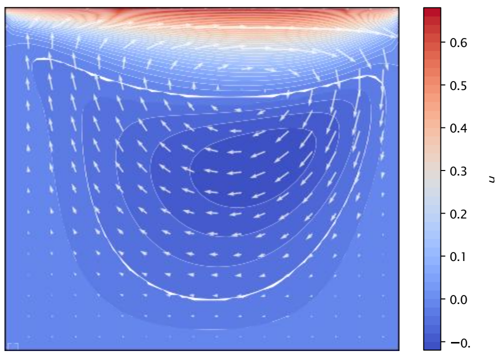
Figure 3: Results of the Kovasznay flow application case: (a) u field determined by the differentiable numerical solver for $\nu = 0.025$ m²/s, (b) predicted vs. true ν in m²/s.

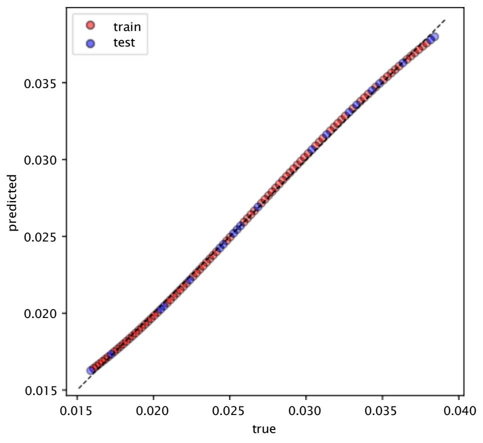
3.3. *Lid-driven cavity*

Data generation. For the well-known lid-driven cavity example, unlike the previous cases, an analytical solution is not available. Therefore, the dataset is generated using the same numerical solver that is integrated into the ML framework. However, to better simulate real-world measurement conditions and account for potential uncertainties, Gaussian noise is added to the generated data, introducing variability that mimics experimental observations. Similar to the previous examples, Re is bounded by setting U to $5 \cdot 10^{-3}\,\mathrm{ms}^{-1}$

Numerical solver. The numerical solver as briefly described in the Kovasznay flow application is also used for the lid-driven cavity case. To keep the computational time during training within reasonable limits, the computational mesh has 51 cells in both the x and y directions. A validation of the numerical results with the reference data of Ghia et al. [18] showed acceptable accordance.

Results. To demonstrate the ability of the differentiable fluid solver to accurately predict the flow, the flow field of the lid-driven cavity of an arbitrary viscosity values is shown in Figure 4 (a). Since no analytical solution exists for this case, the dataset used to train the ML model was generated using the same numerical solver adding 5% Gaussian noise to simulate real-world measurement uncertainties. Figure 4 (b), the ML-predicted viscosity values are compared against the true values. The results form an almost perfect diagonal line, indicating that the model accurately reconstructs viscosity from the noisy input data. This strong correlation suggests that the ML model effectively learns the underlying viscosity distribution, even in the presence of noise, demonstrating its robustness and reliability.





(a) (b)

Figure 4: Results of the lid-driven cavity application case: (a) u field determined by the differentiable numerical solver for $\nu = 0.01$ m²/s, (b) predicted vs. true ν in m²/s.

4. Discussion and conclusion

This study demonstrates the effectiveness of integrating differentiable solvers into ML frameworks for solving inverse problems governed by PDEs. By using automatic differentiation, the proposed approach enables direct optimization of neural network parameters through gradient-based methods, allowing for robust parameter estimation without requiring direct observational data of the target parameters. The method was applied to three different flow scenarios: the viscous Burgers' equation, the Kovasznay flow, and the lid-driven cavity problem. In each case, the neural network successfully learned the underlying viscosity model based on observable velocity fields. The results for the Burgers' equation showed a high degree of agreement between the numerical and analytical solutions, confirming the accuracy of the numerical solver. The Kovasznay flow results exhibited a nearly perfect correlation between the true and predicted viscosity values, although the numerical solver relies on a relatively rough computational resolution. Similarly, in the lid-driven cavity problem, the ML model demonstrated excellent predictive performance, accurately reconstructing viscosity despite the introduction of Gaussian noise to simulate measurement uncertainties. These findings highlight the potential of the proposed differentiable solver-based learning framework for solving inverse problems in fluid dynamics and other PDE-driven systems. Unlike purely data-driven models, this approach ensures that the learned parameters remain consistent with underlying physical principles, improving generalizability and robustness. However, since the current study focuses on the principle functionality of the approach, no measures concerning computational speed have been taken. Although the differentiable solver is able to be executed on graphics processing unit (GPU) along with the neural network training, the backpropagation through the solver by AD is still time consuming. Thus, future work will focus on the use of adjoint methods to compute the gradients effectively. Furthermore, the application of the approach to real-world examples will be explored, focusing on potential use cases such as learning material laws or virtual sensing. Overall, this study confirms that integrating machine learning with differentiable solvers is a powerful and scalable approach for solving inverse problems in computational physics.

Acknowledgments

Stefan Posch would like to acknowledge the financial support of the "COMET Module LEC FFF" within the "COMET - Competence Centers for Excellent Technologies" Programme of the Austrian Federal

Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK), the Austrian Federal Ministry of Labour and Economy (BMAW) and the Provinces of Styria and Tyrol. The COMET Programme is managed by the Austrian Research Promotion Agency (FFG). Marian Staggl and Wolfgang Sanz would like to acknowledge the financial support of the ”ARIADNE” by the ”Take Off” Program, a Research, Technology, and Innovation Funding Program of the Republic of Austria’s Ministry of Climate Action. The ”ARIADNE” Programme is managed by the Austrian Research Promotion Agency (FFG).

References

[1] S. Arridge, P. Maass, O. Ö ktem, and C.-B. Schönlieb, “Solving inverse problems using data-driven models,” *Acta Numerica*, vol. 28, pp. 1–174, 2019.

[2] Q. Lin, N. Allanic, R. Deterre, P. Mousseau, and M. Girault, “In-line viscosity identification via thermal-rheological measurements in an annular duct for polymer processing,” *International Journal of Heat and Mass Transfer*, vol. 182, p. 121988, 2022.

[3] H. Ouaissa, A. Chakib, A. Nachaoui, and M. Nachaoui, “On numerical approaches for solving an inverse cauchy stokes problem,” *Applied Mathematics & Optimization*, vol. 85, no. 1, p. 3, 2022.

[4] A. Kontogiannis, S. V. Elgersma, A. J. Sederman, and M. P. Juniper, “Joint reconstruction and segmentation of noisy velocity images as an inverse navier–stokes problem,” *Journal of Fluid Mechanics*, vol. 944, p. A40, 2022.

[5] M. Raissi, P. Perdikaris, and G. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[6] M. Raissi, A. Yazdani, and G. E. Karniadakis, “Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations,” *Science*, vol. 367, no. 6481, pp. 1026–1030, 2020.

[7] A. D. Jagtap, Z. Mao, N. Adams, and G. E. Karniadakis, “Physics-informed neural networks for inverse problems in supersonic flows,” *Journal of Computational Physics*, vol. 466, p. 111402, 2022.

[8] Y. Bai, X. Chen, C. Gong, and J. Liu, “Impinn: Improved physics-informed neural networks for solving inverse problems,” in *2024 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 189–198, IEEE, 2024.

[9] P. Karnakov, S. Litvinov, and P. Koumoutsakos, “Solving inverse problems in physics by optimizing a discrete loss: Fast and accurate learning without neural networks,” *PNAS Nexus*, vol. 3, p. pgae005, 01 2024.

[10] B. List, L.-W. Chen, and N. Thuerey, “Learned turbulence modelling with differentiable fluid solvers: physics-based loss functions and optimisation horizons,” *Journal of Fluid Mechanics*, vol. 949, p. A25, 2022.

[11] S. Brahmachary and N. Thuerey, “Unsteady cylinder wakes from arbitrary bodies with differentiable physics-assisted neural network,” *Physical Review E*, vol. 109, no. 5, p. 055304, 2024.

[12] E. Franz and N. Thuerey, “Pict: Adaptive gpu accelerated differentiable fluid simulation for machine learning,” in *ICML 2024 Workshop on Differentiable Almost Everything: Differentiable Relaxations, Algorithms, Operators, and Simulators*, 2024.

[13] P. Holl and N. Thuerey, “$\phi_{F\ low}$: Differentiable simulations for machine learning,” in *ICML 2024 Workshop on Differentiable Almost Everything: Differentiable Relaxations, Algorithms, Operators, and Simulators*, 2024.

[14] C. R. Wilke, “A viscosity equation for gas mixtures,” *The journal of chemical physics*, vol. 18, no. 4, pp. 517–519, 1950.

[15] J. D. Cole, “On a quasi-linear parabolic equation occurring in aerodynamics,” *Quarterly of applied mathematics*, vol. 9, no. 3, pp. 225–236, 1951.

[16] E. Hopf, “The partial differential equation $u_t + uu_x = \mu u_x x$,” *Communications on Pure and Applied mathematics*, vol. 3, no. 3, pp. 201–230, 1950.

[17] L. I. G. Kovasznay, “Laminar flow behind a two-dimensional grid,” in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 44, pp. 58–62, Cambridge University Press, 1948.

[18] U. Ghia, K. N. Ghia, and C. Shin, “High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method,” *Journal of computational physics*, vol. 48, no. 3, pp. 387–411, 1982.