Proceedings of the International Conference on Statistics: Theory and Applications (ICSTA'24)
Barcelona, Spain – August 19 - 21, 2024
Paper No. XXX (The number assigned by the OpenConf System)
DOI: TBA

Physics-informed deep learning based on the finite difference method for efficient and accurate numerical solution of partial differential equations

Jiaming Zhang[1], **David Dalton**[1], **Hao Gao**[1], **Dirk Husmeier** [1]
[1]School of Mathematics and Statistics, University of Glasgow
Glasgow G12 8SQ, UK
Emails: Jiaming.Zhang@glasgow.ac.uk; David.Dalton@glasgow.ac.uk
Hao.Gao@glasgow.ac.uk ;Dirk.Husmeier@glasgow.ac.uk

Abstract - In this paper, we present important advancements in the application of Physics-Informed Neural Networks (PINNs) for solving complex partial differential equations (PDEs), which are pivotal in modelling phenomena across various scientific and engineering disciplines. Our approach integrates fourth-order Runge-Kutta (RK4) methods into the loss functions of PINNs, to improve solution accuracy in various benchmark problems such as the 1-D Korteweg–de Vries, 2-D Burgers' and the Navier-Stokes equations. Furthermore, we combine a modified Multi-layer Perceptron (MLP) architecture that noticeably improves the predictive accuracy of PINNs over automatic differentiation (AD) based methods while incurring only a minimal increase in computational costs. Through numerical experiments, our findings demonstrate that the proposed RK4-based loss function and modified MLP architecture offer substantial improvements over AD-based methods, particularly in scenarios characterised by nonlinear and high-dimensional PDEs. This study not only bridges the gap between conventional numerical simulations and deep learning approaches for solving PDEs but also opens new avenues for future research in computational physics and engineering. Our contributions promise to enhance the robustness, efficiency, and applicability of PINNs in tackling a broader range of complex physical problems, marking a significant step forward in the intersection of physical sciences, machine learning and statistics.

Keywords: Physics-informed deep learning, Automatic differentiation, Finite difference method, Navier-Stokes equation

1. Introduction

Partial Differential Equations (PDEs) are foundational to mathematical modelling in many fields, including physics, biology, economics and finance. These equations articulate the dynamics of systems across space and time, capturing phenomena such as heat conduction, fluid motion, and material deformation, which are critical to numerous scientific and engineering applications. For complex scenarios, PDEs tend to be analytically intractable, necessitating numerical solutions. Over recent decades, a variety of numerical strategies have been developed and refined for solving PDEs, including the finite difference (FDM), finite element, finite volume (FVM), and spectral element methods[1], each underpinned by robust theoretical foundations – comprising error estimators and assurances of convergence and stability – and resulting in discretised problems that are amenable to numerical solutions through either sparse linear systems or Newton's method with reliable convergence properties.

On the other hand, recent studies have explored the capacity of deep neural networks (DNNs) to solve PDEs[2]. DNNs are able to fit high dimensional data to overcome the curse of dimensionality and fit data from functional spaces with constrained regularity, such as shock and discontinues [3, 4]. Physics-Informed Neural Networks (PINNs) [5] extend the capabilities of conventional DNNs by integrating underlying physical laws into the training process, thereby enabling the modelling of systems with limited data. By harnessing physics-constrained loss functions, PINNs offer a novel and efficient approach to tackling the computational challenges associated with PDEs, thus bridging the gap between traditional mathematical modelling and contemporary computational methodologies. PINNs usually resolve PDEs by incorporating physics constraints directly into the training loss function through Automatic Differentiation (AD) [5].

Previous work in this area has focused on using first-order global truncation error methods [6] by combining both AD and FDM in the loss function, such as CAN-PINN [7] and finite volume PINN [8]. Recently, Revanth *et al.* [9] proposed a sequential method to train the PINN to reduce further the memory needed in the experiment and Wang *et al.*[10] proposed a modified multi-layer perception (mMLP) architecture that can increase the accuracy of the PINN with only minor increase in the computational cost.

In the present study, we will evaluate the loss function of the PINN by approximating temporal and spatial derivatives

with FDM instead of AD. In particular, the fourth-order Runge-Kutta (RK4) method is used in temporal discretisation. We will demonstrate that this RK4-based PINN can predict more accurate results than conventional AD methods over a range of classical PDEs, including 1-D Korteweg–de Vries, 2-D Burger, and the Navier-Stokes equations.

2. Methodology

2.1. Physics-informed neural networks

Consider a system of PDEs that models a physical system in an open, bounded and connected domain $\Omega \subset \mathbb{R}^n$ for a time interval $[0, T]$:

$$\mathcal{N}_t[\mathbf{u}(\mathbf{x}, t)] + \mathcal{N}_x[\mathbf{u}(\mathbf{x}, t)] = 0, \quad \mathbf{x} \in \Omega, t \in [0, T], \tag{1}$$

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{2}$$

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

where we have denoted this $n-$dimensional PDE system by the temporal derivative $\mathcal{N}_t[\cdot]$ and the general nonlinear differential operator $\mathcal{N}_x[\cdot]$ that can encompass a wide range of nonlinear spatial derivative terms. The primary unknown $\mathbf{u}(\mathbf{x}, t)$ is the solution of the PDE system. The initial state of the system at time $t = 0$ is defined by $\mathbf{u}_0(\mathbf{x})$, and $\mathcal{B}[\cdot]$ is a boundary operator that enforces the desired boundary condition $g(\mathbf{x}, t)$ at the domain boundary $\partial\Omega$, i.e. Dirichlet, Neumann, Robin, or periodic boundary conditions. The operator $\mathcal{B}[\cdot]$ can take the form of either an identity operator or a differential operator.

The idea of the physics-informed neural network (PINN)[5] is approximating $\mathbf{u}(\mathbf{x}, t)$ with a multi-layer perceptron (MLP) $\hat{\mathbf{u}}_{\boldsymbol{\theta}}(\mathbf{x}, t)$, where $\boldsymbol{\theta}$ denotes the network parameters (i.e. weights and biases). Then the PINN can be trained by minimising the following loss function:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_{\text{res}} + \mathcal{L}_{\text{bc}} + \mathcal{L}_{\text{ic}}, \tag{4}$$

where

$$\mathcal{L}_{\text{res}} = \frac{1}{N_{\text{res}}} \sum_{i=1}^{N_{\text{res}}} \left| \mathcal{N}_t[\hat{\mathbf{u}}_{\boldsymbol{\theta}}(\mathbf{x}^i_{\text{res}}, t^i_{\text{res}})] + \mathcal{N}_x[\hat{\mathbf{u}}_{\boldsymbol{\theta}}(\mathbf{x}^i_{\text{res}}, t^i_{\text{res}})] \right|^2 \tag{5}$$

$$\mathcal{L}_{\text{bc}} = \frac{1}{N_b} \sum_{i=1}^{N_{\text{bc}}} \left| \hat{\mathbf{u}}_{\boldsymbol{\theta}}(\mathbf{x}^i_{\text{bc}}, t^i_{\text{bc}}) - g(\mathbf{x}^i_{\text{bc}}, t^i_{\text{bc}})) \right|^2, \tag{6}$$

$$\mathcal{L}_{\text{ic}} = \frac{1}{N_0} \sum_{i=1}^{N_{\text{ic}}} \left| \hat{\mathbf{u}}_{\boldsymbol{\theta}}(\mathbf{x}^i_{\text{ic}}, 0) - \mathbf{u}_0(\mathbf{x}^i_{\text{ic}})) \right|^2, \tag{7}$$

where $\{\mathbf{x}^i_{\text{ic}}\}_{i=1}^{N_{\text{ic}}}$ denotes the initial condition data, $\{(\mathbf{x}^i_b, t^i_b)\}_{i=1}^{N_{\text{bc}}}$ denotes the boundary condition data, and $\{(\mathbf{x}^i_{\text{res}}, t^i_{\text{res}})\}_{i=1}^{N_{\text{res}}}$ are a set of collocation points in a given mesh inside the domain Ω to minimise the PDE residual. The term $\mathcal{L}_{\text{res}}$ imposes penalties when the differential equation is not fulfilled at the collocation points. Furthermore, $\mathcal{L}_{\text{bc}}$ and $\mathcal{L}_{\text{ic}}$ respectively ensure compliance with boundary conditions and initial conditions, thereby maintaining the physical integrity of the solution across the domain. The minimisation of $\mathcal{L}(\boldsymbol{\theta})$ typically leverages stochastic gradient descent, allowing for the sampling of an extensive number of training points. The positions of $\{\mathbf{x}^i_{\text{ic}}\}_{i=1}^{N_{\text{ic}}}$, $\{(\mathbf{x}^i_b, t^i_b)\}_{i=1}^{N_{\text{bc}}}$, and $\{(\mathbf{x}^i_{\text{res}}, t^i_{\text{res}})\}_{i=1}^{N_{\text{res}}}$ are randomised in each iteration of the gradient descent, enhancing the robustness of the learning process. Thus the loss function $\mathcal{L}(\boldsymbol{\theta})$ is minimised as much as possible, ideally approaching zero. This strategy ensures that the neural network accurately represents the underlying physical phenomena by satisfying the differential equations and adhering to the specified initial and boundary conditions.

Traditionally, we can compute the partial derivatives of the predicted solution $\hat{\mathbf{u}}_{\boldsymbol{\theta}}(\mathbf{x}, t)$ in (5) by using neural network's automatic differentiation (AD) feature - *grad* function that takes a function and returns a new function which computes the gradient of the original function, such as JAX [11], which is further equipped with an XLA feature that can increase the computation speed for linear algebra. However, the AD-based PINN, named AD-PINN here, does not preserve the structure of the PDEs, which may affect accuracy and lead to unnecessarily high computational cost [7]. Even more, [12] showed

that AD-PINN's convergent rate is slow compared to numerical methods. An alternative approach is to embed numerical differential operators, i.e. finite difference, in PINN to evaluate the partial derivatives in (5) instead of AD [13, 7]. In this study, we will compare the performance of PINNs using AD and numerical differential operators.

To use the finite difference method (FDM) to approximate PDEs in PINNs when computing the residual loss $\mathcal{L}_{\text{res}}(\boldsymbol{\theta})$, we will need to discretise u in space and in time. Specifically, we use the forward finite difference to discretise first-order spatial derivatives and the central finite difference method for other higher-order spatial derivatives. For temporal derivatives, we use the Euler and the forth-order Runge–Kutta (RK) methods to discretise them. Both Euler and RK methods are numerical integration methods used to approximate the next iteration of the solution in a PDE system.

The numerical schemes for those derivative terms in the PDE loss $\mathcal{L}_{\text{res}}(\boldsymbol{\theta})$ for a scalar-based PDE system in one dimension are listed below.

- Euler method: $u(x, t + \Delta t) = u(x, t) - \Delta t \mathcal{N}_x[u(x, t)]$
- RK4 method: $u(x, t+\Delta t) = u(x, t)+\frac{\Delta t}{6}(k_1+2k_2+2k_3+k_4)$, where $k_1 = -\mathcal{N}_x[u(x, t)]$, $k_2 = -\mathcal{N}_x\left[u\left(x, t + \frac{\Delta t}{2}\right) + \frac{\Delta t k_1}{2}\right]$, $k_3 = -\mathcal{N}_x\left[u\left(x, t + \frac{\Delta t}{2}\right) + \frac{\Delta t k_2}{2}\right]$, $k_4 = -\mathcal{N}_x[u(x, t + \Delta t) + \Delta t k_3]$
- 1^{rd} order forward finite difference: $u_x = \frac{u_(x+\Delta x,t)-u_(x,t)}{\Delta x}$
- 2^{rd} order central finite difference: $u_{xx} = \frac{u_(x-\Delta x,t)-2u_(x,t)+u_(x+\Delta x,t)}{\Delta x^2}$
- 3^{rd} order central finite difference: $u_{xxx} = \frac{u_(x+2\Delta x,t)-2u_(x+\Delta x,t)+2u_(x-\Delta x,t)-u_(x-2\Delta x,t)}{2\Delta x^3}$

2.2. Multi-layer perceptron

Wang *et all.* [10, 14] have proposed a novel architecture, referred to as modified MLP (mMLP), which was found to outperform conventional MLPs in various PINN benchmark studies by a factor of 50-100x in the predictive accuracy, subject to only a slight increase in the computational cost and memory requirements. In mMLP, the inputs $\mathbf{X}$ are embedded into a high-dimensional feature space via two encoders $\mathbf{U}$ and $\mathbf{V}$, then a point-wise multiplication operation is used to update hidden layers. The architecture of mMLP can be described as following,

$$\mathbf{U} = \sigma(\mathbf{X}\mathbf{W}^1 + \mathbf{b}^1), \quad \mathbf{V} = \sigma(\mathbf{X}\mathbf{W}^2 + \mathbf{b}^2) \tag{8}$$

$$\mathbf{H}^{(1)} = \sigma(\mathbf{X}\mathbf{W}^{z,1} + \mathbf{b}^{z,1}) \tag{9}$$

$$\mathbf{Z}^{(k)} = \sigma(\mathbf{H}^{(k)}\mathbf{W}^{z,k} + \mathbf{b}^{z,k}), \quad k = 1, \ldots, L \tag{10}$$

$$\mathbf{H}^{(k+1)} = (1 - \mathbf{Z}^{(k)}) \odot \mathbf{U} + \mathbf{Z}^{(k)} \odot \mathbf{V}, \quad k = 1, \ldots, L \tag{11}$$

$$\hat{\mathbf{u}}_{\boldsymbol{\theta}}(x) = \mathbf{H}^{(L+1)}\mathbf{W} + \mathbf{b} \tag{12}$$

where $\mathbf{W}$ and $\mathbf{b}$ denote the weights and biases, respectively, $\odot$ denotes point-wise multiplication, $\sigma : \mathbb{R}^M \to \mathbb{R}^M$ denotes a nonlinear activation function, and L is the number of hidden layers. All the parameters of mMLP can be denoted as

$$\boldsymbol{\theta} = \{\mathbf{W}^1, \mathbf{b}^1, \mathbf{W}^2, \mathbf{b}^2, (\mathbf{W}^{z,l}, \mathbf{b}^{z,l})_{l=1}^{L}, \mathbf{W}, \mathbf{b}\}. \tag{13}$$

3. Results

In this section, we will demonstrate the predictive accuracy of the proposed RK4-based derivative approximation (RK4-PINN) when evaluating the PDE loss function in PINNs against the Euler method (Euler-PINN) and the AD method (AD-PINN). Our evaluation includes rigorous numerical tests on solving quintessential equations like the 1-D Korteweg–De Vries (KdV), 2-D Burgers, and 2-D incompressible Navier-Stokes. Notably, while these benchmark problems are readily addressable using classical numerical methods, they have posed significant challenges to PINNs since their resurgence in the work of Raissi *et al* [5]. Our discussion will highlight the improvements and advancements made possible by integrating RK4 into the loss function of PINNs, underscoring its potential to overcome previously observed limitations.

In all the experiments, we will utilise the mMLP architecture outlined in Section 2.2, employing a hyperbolic tangent activation function. Moreover, all the networks are initialised with the Glorot normal scheme [15] and trained with the Adam optimiser [16], a stochastic gradient descent method. We set the learning rate 0.001 with an exponential decay with a decay rate of 0.9 every 5000 training iterations, and no additional regularisation techniques are employed. Following [17, 9], we further incorporate time-marching techniques to alleviate optimisation challenges. Specifically, the temporal domain of interest, $[0, T]$, will be divided into successive sub-domains such as $[0, \Delta t]$, $[\Delta t, 2\Delta t]$, ..., $[T - \Delta t, T]$. We then sequentially train neural networks to approximate the solution within each sub-domain, using the predictions from the previously trained network as the initial conditions for the subsequent sub-domain. This methodology allows the resulting PINNs to generate accurate predictions for any given query point across the temporal domain. All hyper-parameter settings and computational costs are list in Table 1, and the L^2 norm errors from trained PINNs are summarised in Table 2. All experiments were run on the same machines that are equipped with one Intel(R) Xeon(R) Gold 6238R CPU and 2 NVIDIA RTX A6000 graphics cards.

Table 1: Network architectures and computational cost reported timings for each benchmark employed in this work.

Case	Methods	# Time windows	# Hidden layers	# Neurons	N_t	N_x	# Iterations	Training rate (iter/sec)
KdV	RK4	5	5	32	201	512	5×10^5	102.58
	Euler	5	5	32	201	512	5×10^5	105.46
	AD	5	5	32	201	512	5×10^5	99.26
2-D Burgers	RK4	10	6	32	32	101	5×10^5	10.13
	Euler	10	6	32	32	101	5×10^5	11.06
	AD	10	6	32	32	101	5×10^5	9.64
Navier-Stoke	RK4	10	6	64	32	256	5×10^5	9.32
	Euler	10	6	64	32	256	5×10^5	9.98
	AD	10	6	64	32	256	5×10^5	8.95

Table 2: L^2 error for each benchmark employed in this work. The two right-most columns show the percentage improvement of RK4-PINN over each of the other two methods, respectively.

		AD-PINN	Euler-PINN	RK4 -PINN	RK4-PINN vs AD-PINN	RK4-PINN vs Euler-PINN
KdV	u	91.7e-04	33.2e-04	6.47e-04	92.9%	80.5%
2-D Burgers	u	9.62e-02	7.39e-02	4.03e-02	58.1%	45.5%
	v	10.2e-02	6.39e-02	3.57e-02	65%	44.1%
Navier-Stoke	u	9.77e-02	6.73e-02	3.90e-02	60.1%	42.1%
	v	9.31e-02	6.39e-02	3.61e-02	61.2%	43.5%
	w	10.3-02	9.41e-02	6.12e-02	40.6%	35%

3.1. Korteweg–de Vries equation

We use Korteweg–de Vries (KdV) equation as our first example. This is designed to showcase the adeptness of our proposed methodology in managing partial differential equations that entail higher-order derivatives. The KdV equation describes the evolution of long one-dimensional waves in many physical settings. It characterises the propagation of long, unidirectional waves in various contexts, notable among which are shallow-water waves under the influence of gentle nonlinear restoring forces, the propagation of long internal waves in an ocean with stratified density, ion-acoustic waves within a plasma,

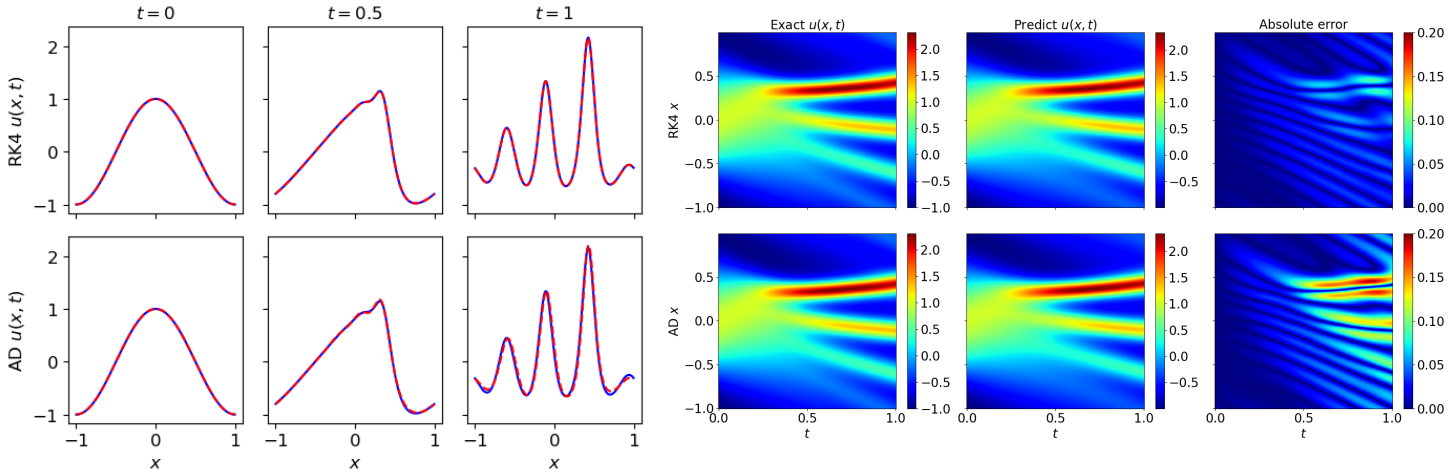

Fig. 1: *1-D KdV equation*: *Left:* Comparison of the predicted and reference solutions corresponding to the three temporal snapshots at $t = 0, 0.5, 1.0$. *Right:* Predicted velocity versus the corresponding reference solution at $t = 1$

and acoustic waves propagating through a crystal lattice [5]. The KdV equation with periodic boundary conditions is given by

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + 0.0025\frac{\partial^3 u}{\partial x^3} = 0, \quad t \in [0, 1], x \in [-1, 1] \tag{14}$$

and the initial condition is $u(x, 0) = \cos(\pi x)$.

Our objective is to learn the corresponding solution up to a time $t = 1$. A comprehensive visual evaluation of the predicted solution is detailed in Figure 1. Specifically, we offer a side-by-side comparison of the reference obtained from [5] and the predicted solutions at $t = 0, 0.5, 1.0$. The comparison reveals that the predictions from AD-PINN and RK4-PINN methods are in good agreement with the reference solutions. However, we obverse that in the left panel of Figure 1 at $t = 1$, the solution from AD-PINN deviates from the reference solution near the turning points, but not in RK4-PINN. RK4-PINN achieves the least L^2 error of 6.47e-4 compared to AD-PINN and Euler-PINN, which are 9.17e-3 and 3.32e-3, respectively.

3.2. 2-D Burgers' equation

To further study RK4 method, we now use the 2-D Burgers' equation as the second example due to its widespread application in various branches of applied mathematics, including but not limited to fluid mechanics, nonlinear acoustics, gas dynamics, and traffic flow [18]. Burgers' equation can be deduced from Navier-Stokes equations by eliminating the pressure gradient component. When dealing with minimal viscosity parameters, Burgers' equation may result in the development of shocks, which poses significant challenges for resolution through conventional numerical techniques. The Burgers' equation with two spatial dimensions is given by

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \nu \Delta \mathbf{u}, \quad t \in [0, 1], x \in [0, 1], y \in [0, 1], \tag{15}$$

where $\mathbf{u} = (u, v)$ is the flow velocity field, $\nu = 0.01/\pi$, $\nabla := (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$ is the gradient operator, and $\Delta := (\frac{\partial^2}{\partial x^2}, \frac{\partial^2}{\partial y^2})$ is the Laplace operator. The initial condition is $u(x, y, 0) = \sin(2\pi x)\sin(2\pi y)$, $v(x, y, 0) = \sin(\pi x)\sin(\pi y)$ with Dirichlet boundary condition that u and v on the boundaries are equal to 0.

To obtain the reference solution of the 2-D Burgers' equation, we have employed numerical techniques for differentiation and integration to approximate the solution of the 2-D Burgers' equation. Specifically, we laid out a grid consisting of 101 uniformly spaced nodes in each spatial direction, both x and y. To estimate the spatial derivatives required by the governing equations, we utilised a sixth-order compact finite difference scheme, known for its high accuracy in capturing spatial

variations. For temporal integration, we implemented a fourth-order Runge-Kutta scheme, acknowledged for its stability and precision. The integration process was carried out with a finely selected time step of 10^{-5}, ensuring the detailed resolution of the solution's evolution over time.

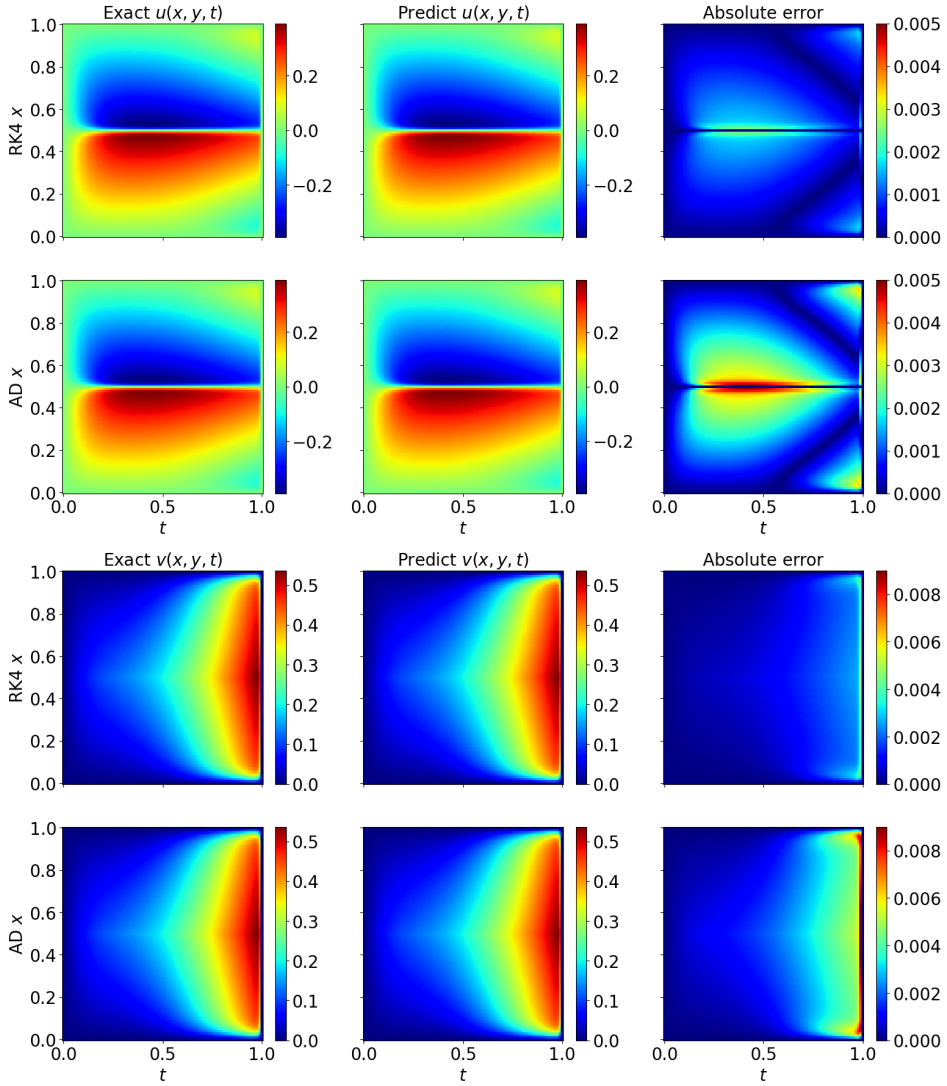

Fig. 2: *2-D Burgers' equation:* Predicted velocity versus the corresponding reference solution at $t = 1$

Figure 2 presents the velocity field at $t = 1$ for both RK4 and AD methods. We can see that all latent variables of interest are in good agreement with their corresponding reference solutions for the RK4 method, yielding an error of 4.03e-02 and 3.57e-02 for u and v respectively. For the Euler and AD method, the results are displayed in Table 2. We can see that the RK4 method has increased computational speed by 5% over the AD method, and improved accuracy in terms of the L^2 error by 63%.

3.3. Navier–Stokes equation

To underscore the capability of our proposed method in addressing chaotic dynamical systems, we turn our attention to a classical two-dimensional decaying turbulence example in a square domain with periodic boundary conditions, which is

modelled by the incompressible Navier-Stokes equations using the velocity-vorticity formulation [19], that is

$$\frac{\partial w}{\partial t} + \mathbf{u} \cdot \nabla w = \frac{1}{\mathrm{Re}} \Delta w, \quad t \in [0, 1], x \in [0, 2\pi], y \in [0, 2\pi], \tag{16}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{17}$$

$$w(0, x, y) = w_0(x, y), \tag{18}$$

where $\mathbf{u} = (u, v)$ is the flow velocity field, $w = \nabla \times \mathbf{u}$ is the vorticity and Re is the Reynolds number with Re = 100.

Figure 3 presents the predicted vorticity field at $t = 1$ using RK4-PINN and AD-PINN, along with the reference solution obtained from [19]. We can see that all latent variables of interest are in good agreement with the reference solution, yielding an error of 6.12e-2 for RK4-PINN and 10.3e-2 for AD-PINN. A 41% error reduction is achieved in RK4-PINN compared to AD-PINN. Similar results can be found for u and v, not shown due to the page limit. Errors in predicted u, v and w using AD-PINN, Euler-PINN and RK4-PINN are summarised in Table 2. These results highlight the effectiveness of the proposed RK4 method, successfully enabling the PINNs model to accurately capture such complicated turbulent flow without any training data.

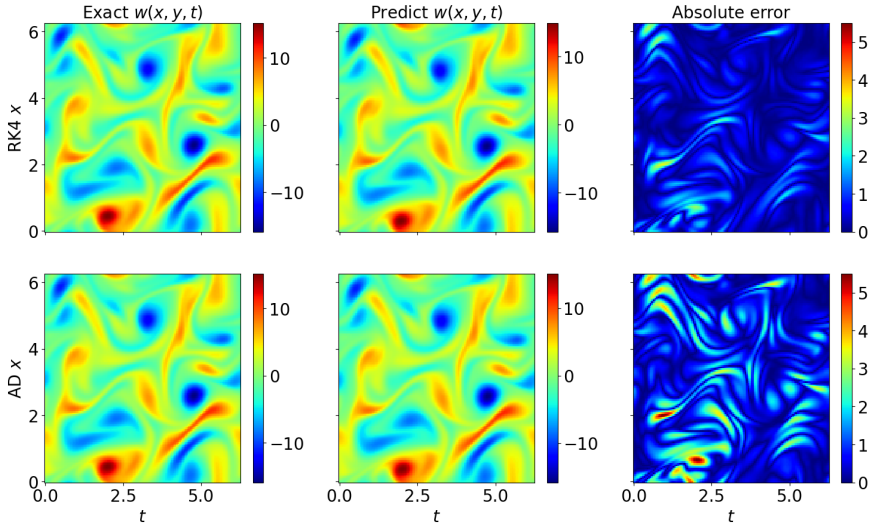

Fig. 3: *Navier-Stokes equation:* Predicted vorticity versus the corresponding reference solution at $t = 1$.

4. Conclusion

This paper has explored the forefront of enhancing PINNs through the integration of fourth-order RK4 methods into their loss functions using a modified Multi-layer Perceptron architecture proposed by Wang et al.[10]. We have demonstrated improved accuracy and computational efficiency of this new PINN approach across a series of benchmark partial differential equations, such as the Korteweg–de Vries, 2-D Burgers' equations and the Navier-Stokes equation. This study underscores the viability of PINNs by combining numerically approximated derivatives in capturing the nuances of physical phenomena, making strides toward a future where machine learning and computational science are seamlessly integrated. Through rigorous testing, we have validated the utility of our proposed methodologies and laid a foundation for their application to an even broader array of complex problems in the future.

Our future work will focus on the estimation of the PDE parameters and their posterior uncertainty quantification from limited observed data. Repeated adaption of PDE parameters as part of an iterative optimisation or sampling routine would be computationally onerous when performed with established numerical techniques like finite element methods, and we will quantify by how much that can be improved with the novel physics-informed machine learning methods presented in the present work.

Acknowledgements

This work has been supported by China Scholarship Council (CSC) and a fee waiver from the University of Glasgow, by the Engineering and Physical Sciences Research Council (EPSRC) of the United Kingdom, grant reference number EP/T017899/1, and by the British Heart Foundation ((PG/22/10930).

References

[1] K. W. Morton and D. F. Mayers, *Numerical solution of partial differential equations: an introduction*. Cambridge university press, 2005.

[2] W. E and B. Yu, "The deep ritz method: A deep learning-based numerical algorithm for solving variational problems," 2017.

[3] F. Bach, "Breaking the curse of dimensionality with convex neural networks," *Journal of Machine Learning Research*, vol. 18, no. 19, pp. 1–53, 2017.

[4] J. Han, A. Jentzen, and W. E, "Solving high-dimensional partial differential equations using deep learning," *Proceedings of the National Academy of Sciences*, vol. 115, no. 34, pp. 8505–8510, 2018.

[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] Z. Xiang, W. Peng, W. Zhou, and W. Yao, "Hybrid finite difference with the physics-informed neural network for solving pde in complex geometries," *arXiv preprint arXiv:2202.07926*, 2022.

[7] P.-H. Chiu, J. C. Wong, C. Ooi, M. H. Dao, and Y.-S. Ong, "Can-pinn: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method," *Computer Methods in Applied Mechanics and Engineering*, vol. 395, p. 114909, May 2022.

[8] T. Praditia, M. Karlbauer, S. Otte, S. Oladyshkin, M. V. Butz, and W. Nowak, "Finite volume neural network: Modeling subsurface contaminant transport," 2021.

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

[10] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient pathologies in physics-informed neural networks," 2020.

[11] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.

[12] Z. Fang, "A high-efficient hybrid physics-informed neural networks based on convolutional neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 10, pp. 5514–5526, 2022.

[13] K. L. Lim, R. Dutta, and M. Rotaru, "Physics informed neural network using finite difference method," in *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1828–1833, IEEE, 2022.

[14] S. Wang, H. Wang, and P. Perdikaris, "On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 384, p. 113938, 2021.

[15] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, JMLR Workshop and Conference Proceedings, 2010.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] C. L. Wight and J. Zhao, "Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed neural networks," *arXiv preprint arXiv:2007.04542*, 2020.

[18] C. Basdevant, M. Deville, P. Haldenwang, J. Lacroix, J. Ouazzani, R. Peyret, P. Orlandi, and A. Patera, "Spectral and finite difference solutions of the burgers equation," *Computers Fluids*, vol. 14, no. 1, pp. 23–41, 1986.

[19] S. Wang, S. Sankaran, and P. Perdikaris, "Respecting causality is all you need for training physics-informed neural networks," *arXiv preprint arXiv:2203.07404*, 2022.