Applications of physics informed neural operators

IOP Publishing Mach. Learn.: Sci. Technol. 4 (2023) 025022 https://doi.org/10.1088/2632-2153/acd168

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
14 December 2022

REVISED
13 April 2023

ACCEPTED FOR PUBLICATION
28 April 2023

PUBLISHED
18 May 2023

PAPER

Applications of physics informed neural operators

Shawn G Rosofsky[1,2,3,*], Hani Al Majed[1,3,4] and E A Huerta[1,2,5,*]

[1] Data Science and Learning Division, Argonne National Laboratory, Lemont, IL 60439, United States of America
[2] Department of Physics, University of Illinois at Urbana-Champaign, Urbana, IL 61801, United States of America
[3] NCSA, University of Illinois at Urbana-Champaign, Urbana, IL 61801, United States of America
[4] Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801, United States of America
[5] Department of Computer Science, The University of Chicago, Chicago, IL 60637, United States of America
[*] Authors to whom any correspondence should be addressed.

E-mail: shawngr2@illinois.edu and elihu@anl.gov

Keywords: physics informed deep learning, surrogate model, neural operators

Abstract

We present a critical analysis of physics-informed neural operators (PINOs) to solve partial differential equations (PDEs) that are ubiquitous in the study and modeling of physics phenomena using carefully curated datasets. Further, we provide a benchmarking suite which can be used to evaluate PINOs in solving such problems. We first demonstrate that our methods reproduce the accuracy and performance of other neural operators published elsewhere in the literature to learn the 1D wave equation and the 1D Burgers equation. Thereafter, we apply our PINOs to learn new types of equations, including the 2D Burgers equation in the scalar, inviscid and vector types. Finally, we show that our approach is also applicable to learn the physics of the 2D linear and nonlinear shallow water equations, which involve three coupled PDEs. We release our artificial intelligence surrogates and scientific software to produce initial data and boundary conditions to study a broad range of physically motivated scenarios. We provide the source code, an interactive website to visualize the predictions of our PINOs, and a tutorial for their use at the Data and Learning Hub for Science.

1. Introduction

The description of physical systems has a common set of elements, namely: the use of fields (electromagnetic, gravitational, etc) on a given spacetime manifold, a geometrical interpretation of these fields in terms of the spacetime manifold, partial differential equations (PDEs) on these fields that describe the change of a system over spacetime, and an initial value formulation of these equations with suitable boundary conditions (BCs) [1]. Given that the evolution of physical fields over spacetime may be naturally expressed in terms of PDEs, a plethora of numerical methods have been developed to accurately and rapidly solve these class of equations [2].

In time, and even with the advent of extreme scale computing, some physical systems have become increasingly difficult to model, e.g. multi-scale and multi-physics systems that combine disparate time and spatial scales, and which demand the use of subgrid-scale precision to accurately resolve the evolution of physical fields. This is a well known problem in multiple disciplines, including general relativistic simulations [3–5], weather forecasting [6], *ab initio* density functional theory simulations [7], among many other computational grand challenges.

The realization that large scale computing resources are finite and will continue to be oversubscribed [8, 9], has compelled scientists to explore novel approaches to address computational bottlenecks in scientific software [10]. Some approaches include rewriting modules of software stacks to leverage GPUs, leading to significant speedups [11–13]. Other contemporary approaches have harnessed advances in machine learning to accelerate specific computations in software modules [14], while others have developed entirely new

solutions by combining GPU-accelerated computing and novel signal processing tools that have at their core machine learning or artificial intelligence applications [15–21].

The confluence of AI and advanced computing aims to enable breakthroughs in science and engineering that would otherwise remain unfeasible with traditional approaches. Furthermore, AI surrogates aim to capture known knowledge, and first principles to steer AI learning in the right direction, and then refine AI surrogates performance and predictions by using experimental scientific data. In time, it is expected that by exposing AI to detailed simulations, first principles and experimental data, AI surrogates will capture the nonlinear behavior of experimental phenomena and guide the planning, automation and execution of new experiments, leading to breakthroughs in science and engineering. These approaches exhibit great promise when coupled with robotics labs [22].

In this article we contribute to the construction of AI surrogates by demonstrating their application to solve a number of PDEs that are ubiquitous in physics, and which have not been presented before in the literature. Specifically, we have three levels of applicability of these new problems. First, we performed sanity checks with simple PDEs to verify that our model behaves as expected. Second, we tested new numerically challenging cases to assess the applicability of physics informed neural operators (PINOs) under such conditions, which include non-constant coefficients, coupled PDEs, and shocks. Third, we applied PINOs to coastal and tsunami modeling with the 2D linear and nonlinear shallow water equations. Beyond these original contributions, we also provide an end-to-end framework that unifies initial data production, construction of BCs and their use to train, validate and test the performance and reliability of AI surrogates. These activities aim to create FAIR findable, accessible, interoperable and reusable and AI-ready datasets and AI models [23–26].

In the following sections we introduce key concepts and ideas that will facilitate the understanding and use of physics-inspired neural operators (PINOs) to solve PDEs [27]. This article is organized as follows. In section 2 we introduce the AI tools we use to learn the physics of PDEs. We describe our methods and approaches to create PINOs in section 3. We summarize the PDEs we consider in this study in section 4, and present a direct comparison between numerical solutions of these PDEs and predictions from our PINOs in section 5. We describe future directions of work in section 6.

2. Modeling physics with artificial neural networks

2.1. Physics informed neural networks (PINNs)

PINNs provide a method of using known physical laws to predict the results of various physical systems at high accuracy [28–32]. These methods estimate the results for a given physical system consisting of a PDE, initial conditions (ICs), and BCs by minimizing constraints in the loss function. PINNs utilize the automatic differentiation of deep learning frameworks to compute the derivatives of the PDE to compute the residual error. Despite the success of PINNs, their inability to produce results for different ICs prevents them from being useful surrogate models.

2.2. Neural operators

Neural operators use neural networks to learn operators rather than single physical systems. In our case, PDEs are the operator these networks try to learn. Specifically, we provide neural operators with input fields $\mathcal{A}$ that are composed of coordinates and relevant data such as coefficient fields, ICs, and BCs. Neural operators then output the solutions of that operator at those coordinates which we will denote as $\mathcal{U}$. We can mathematically represent the neural operator $\mathcal{G}_\theta$ as a mapping between the input fields $\mathcal{A}$ and the output fields $\mathcal{U}$ as

$$\mathcal{G}_\theta : \mathcal{A} \to \mathcal{U} \tag{1}$$

where θ are the weights of the neural operator [33].

There are various types of neural operators that have been studied in recent works such as DeepONets, physics informed DeepONets, low-rank neural operators, graph neural operators, multipole graph neural operators, Fourier neural operators (FNO), and PINOs [27, 33–38]. These networks have all illustrated their ability to reproduce the results of operators much faster than computing with the operator directly. For a more detailed look at neural operators, we refer the reader to [33], which provides rigorous definitions of neural operators and examines a large number of different neural operators.

2.3. PINOs

PINOs are a variation of neural operators that incorporate knowledge of physical laws into their loss functions [27]. PINOs have been shown to reproduce the results of operators with remarkable accuracy.

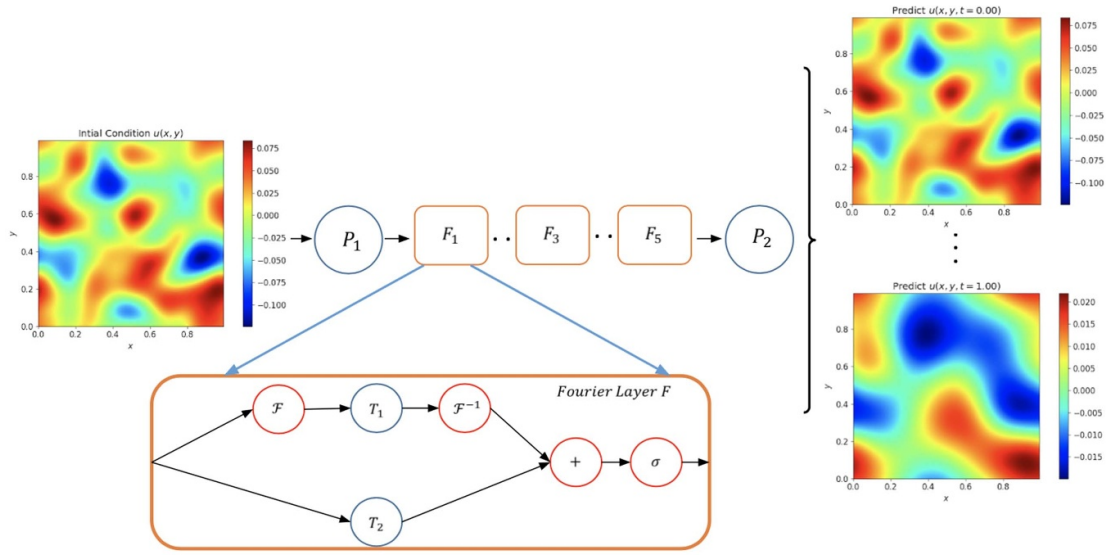

Figure 1. Neural network architecture. The top panel shows the architecture of our neural operators, whereas the inset in the bottom panel shows the structure of one of the five Fourier layers, $\{F_1, \ldots, F_5\}$, we use in our PINOs. The top left panel shows the data we feed into our models, labelled as 'Initial Condition $u(x, y)$'. This input data are initially lifted into a higher dimension representation by the neural network, P_1. Thereafter, we apply a series of updates that consist of non-local integral operators and nonlinear activation functions, σ, shown in the inset labelled as '*Fourier Layer F*'. Eventually, the neural network P_2 projects back the updates, producing the output shown on the right panels, which describe the time evolution of the system. In the inset, T_1 represents a linear transform, T_2 a local linear transform, σ a nonlinear activation function. $\mathcal{F}$ and $\mathcal{F}^{-1}$ stand for Fourier transform and inverse Fourier transform, respectively.

They employ the FNO architecture which applies a fast Fourier transform (FFT) to the data and applies its fully connected layers in Fourier space before performing an inverse FFT back to real space [38]. Moreover, this architecture has demonstrated the ability to perform zero-shot super-resolution, predicting on higher resolution data having only seen low resolution data [33, 38]. Figure 1 illustrates the FNO architecture that we use in this study.

PINOs improve upon the FNO architecture by adding physics information such as PDEs, ICs, BCs, and other conservation laws. By including the violation of such laws into the loss function, the network can learn these laws in addition to the data. Rather than using automatic differentiation, these networks use Fourier derivatives to compute the derivatives for the PDE constraints as automatic differentiation is very memory intensive for this type of architecture. This physics knowledge enables the network to learn operators faster and with less training data [39].

3. Methods

Here we describe the approach we followed to design, train, validate and test our PINOs, and then how we quantified their accuracy by comparing our predictions with actual numerical solutions of PDEs we consider in this study.

3.1. Initial data

The first step in this process was to generate initial data that matches the BCs of the problem. This was done using the Gaussian random fields method in a similar fashion to [27] where the kernel was transformed into Fourier space to match our periodic BCs. We employed a general Matern kernel, but discovered that the PINOs performed better when the smoothness parameter was set to infinity. In this limit, this Matern kernel becomes the radial basis function kernel as used in [34] defined as $k_l(x_1, x_2) = \exp\left(-\|x_1 - x_2\|^2 / 2l^2\right)$, where l is the length scale of typical spatial deviations in the data. For this work, we set $l = 0.1$ for all cases to provide features at our desired spatial scale except when explicitly stated otherwise. A number of these random fields were produced for each of the test problems described section 4.

During this work, we found that the magnitude of the training data affected the results of the training even in linear problems. In particular, the models seem to have difficulty with higher magnitude initial data, especially when the data had a magnitude greater than 1. We found that by reducing the magnitude of the initial data, we could improve our results. We believe this is attributed to the highly nonlinear nature of the neural network model. Other neural network models also experience similar effects and therefore normalize

their data to improve performance. Thus, we were careful in selecting the magnitude of our input fields during this study.

3.2. BCs

For our BCs, we used periodic BCs in all cases. This BC is good for problems with significant symmetry over long length scales or for cases where the boundary is sufficiently far from the region of interest. We are particularly focused on the latter case where the BC does not significantly impact the problem. Moreover, the neural network architecture implements periodic BCs by default via FFTs. This periodicity of a certain dimension can be removed by zero padding for said dimension before feeding it to the neural network. We employed this padding in the time dimension with a length of $l_{\text{pad}} = 5$ for all cases except the 1D wave equation, which is periodic in time for a time interval of $t = 1$.

Although we do not do so in this work, one could also use this technique for PDEs with non-periodic BCs by zero padding the spatial dimensions of the data. We could then add an additional loss term for the BC to the loss in equation (2) to describe our desired BC. We describe how to model similar terms in more detail in section 3.4.

3.3. Training data generation

To generate training data, we took the random IC fields that we had previously generated and evolved them in space and time. Specifically, we evolve each of these equations in time with RK4 time stepping starting at $t = 0$ until $t = 1$ with the timestep δt varying depending on the problem. To compute spatial derivatives, we employed a finite difference method with fourth order central difference for most cases. The lone exception was the inviscid Burgers equation in 2D which required a shock capturing method. Therefore, we employed a finite volume method (FVM) with local Lax–Friedrichs fluxes and MP5 reconstruction.

3.4. Training approach

We set up the problem as follows. We are given some space and time coordinates as well as the ICs at those coordinates. Our objective is to compute the solution at each given space and time coordinate from the initial data. To train the network to reproduce the simulated results, we employ multiple losses to ensure the network properly reproduces the correct loss. These losses are the data loss $\mathcal{L}_{\text{data}}$, the physics loss $\mathcal{L}_{\text{phys}}$, and the IC loss $\mathcal{L}_{\text{IC}}$. We do not include a loss term for the BC here because the FNO architecture of the PINO assures the desired periodic BCs as long as those dimensions are not padded.

$\mathcal{L}_{\text{data}}$ attempts to fit the model predictions directly to the training data. This loss is computed via the relative mean squared error (MSE) between the training data and the network outputs. This relative MSE is computed by dividing the MSE of the predictions by the norm of the true values. For cases with multiple equations where the outputs vary in magnitude such as the linear and nonlinear shallow water equations, care must be taken to ensure that each of those outputs contributes equally to $\mathcal{L}_{\text{data}}$. Therefore, we compute the relative MSE for each of the output fields separately before combining them together.

$\mathcal{L}_{\text{phys}}$ ensures that the PINO predictions obey known physical laws such as PDEs or more general conservation laws such as conservation of mass, energy, or momentum. We define the loss as the MSE between the violation of the physical law and the value of said law if it was perfectly satisfied which is 0 in most cases. Again, one must be careful in cases like the linear and nonlinear shallow water equations with multiple physical laws whose violations differ significantly in their magnitude. In such cases, compute the physical violations separately and multiply each term by some weight before combining them and computing the MSE of the combined term to obtain $\mathcal{L}_{\text{phys}}$.

$\mathcal{L}_{\text{IC}}$ allows the PINO to learn how the input IC given to the network is the value of the output at that point at $t = 0$. Moreover, by minimizing $\mathcal{L}_{\text{IC}}$, $\mathcal{L}_{\text{phys}}$ more easily converges to the correct solution for cases where the physical law is a PDE. We defined $\mathcal{L}_{\text{IC}}$ as the relative MSE between the PINO prediction at $t = 0$ and the IC fed into the network. There were no cases where the ICs differed significantly in magnitude for the PDEs since for the linear and nonlinear shallow water equations, the velocity fields were taken to be zero at $t = 0$ and were not fed into the initial data.

To combine these terms into the total loss $\mathcal{L}_{\text{tot}}$, we perform a weighted sum defined as

$$\mathcal{L}_{\text{total}} = w_{\text{data}}\mathcal{L}_{\text{data}} + w_{\text{phys}}\mathcal{L}_{\text{phys}} + w_{\text{IC}}\mathcal{L}_{\text{IC}}\,, \tag{2}$$

where w_{data} is the data weight, w_{phys} is the physics weight, and w_{IC} is the IC weight. We varied these values between different cases and even during the training. Typically, we would set w_{data} to be 5 or 10, w_{phys} to be 1 or 2, and w_{IC} to be 5 or 10.

We emphasize that we select the weights in the loss function through a process of trial and error rather than a sophisticated hyperparameter optimization process. This is typically the case with PINNs as well. If we

take some limits, we can understand why this is the case. If w_{phys} is very high, the PINO will try to minimize $\mathcal{L}_{\text{phys}}$ at the cost of the other parameters. One solution that satisfies the most PDEs is if the output field is zero for all space and time coordinates. Therefore, we add the IC loss $\mathcal{L}_{\text{IC}}$ to ensure that $\mathcal{L}_{\text{phys}}$ evolves the correct initial data. However, if $\mathcal{L}_{\text{IC}}$ is too large, the output field is correct at $t = 0$, but it does not evolve with time. Thus, we used a process of trial and error to determine the correct weights.

For the training itself, we used the `PyTorch` framework [40]. We employed an Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$ with an initial learning rate of 0.001. To fine tune the latter training steps, we employed a multistep scheduler to reduce the learning rate at several intervals throughout the training with a gamma value of 0.5. The specific number of epochs and the epoch decay milestones vary between the different test cases. A common setup that was used from several of the 2D cases was to train for 150 epochs and put the scheduler milestones at [25, 50, 75, 100, 125, 150] epochs. To further improve the performance, we would use the checkpoints after 150 epochs of training and retrain again after for this same duration. In between restarting from these checkpoints, we would sometimes modify the weights of the various training losses if we observed some losses were lagging behind the others.

3.5. Model architecture

As noted in 2.3, the model uses the FNO architecture that is shown in figure 1. We used a Gaussian error linear units for our nonlinearity for all cases [41]. We note that for physics informed deep learning, we require the nonlinear activation function to have a non-zero second derivative, ruling some commonly used activation functions such as the rectified linear unit [30].

The FNO architecture can be described by the widths, modes, and number of layers. The widths and modes are given as arrays representing the width and modes for each entry. The size of the array gives the number of layers and each entry corresponds to a different layer. We used four layers in all cases. For the 1D cases, we used the width $= [16, 24, 24, 32]$ and modes $= [15, 12, 9, 9]$. For the 2D cases, we used width $= [64, 64, 64, 64]$ and modes $= [8, 8, 8, 8]$.

In addition, the network ends with a fully connected layer. We selected a width of 128 for the fully connected layer for all cases.

3.6. Performance quantification

To quantify the performance, we ran PINO on the test dataset, then calculated the MSE of their predictions with the test data and the MSE of the physics loss that accompanied violations of physical laws. The test dataset is composed of approximately 10% of the simulations produced from the random initial data that was separated from the rest of the data to ensure that the PINO did not train on it.

4. Test problems

We use PINOs to learn nine different PDEs. We consider the wave equation in 1D and 2D to demonstrate that our methods produce accurate and reliable results. We also present results for the 1D Burgers equation, which was used in [27] to quantify the performance of PINOs to learn PDEs. We then put our methods at work to solve a variety of PDEs with different levels of complexity.

Specifically, many past works including [27, 33, 36–38] investigating using nonlinear neural operators study only three cases, the 1D Burgers equation, 2D Darcy Flow, and the 2D Navier Stokes equations. While these cases provide a good way to compare neural operators to each other, they limited the applicability of neural operators to other problems. Moreover, the aforementioned cases fail to include various physical and numerical phenomena such as shocks, coupled PDEs, and non-constant coefficients. Therefore, we chose a wide array of linear and nonlinear PDEs to evaluate the PINO models and isolate the places where they may have difficulty.

4.1. Wave equation 1D

Our first test was the wave equation in 1D with periodic BCs. This is a computationally simple PDE that is second order in time and models a variety of different physics phenomena. The equation for the evolved field $u(x, t)$ takes the form

$$\begin{aligned} &u_{tt} + c^2 u_{xx} = 0, \\ &\quad u(x, 0) = u_0(x), \\ &x \in [0, 1), t \in [0, 1], \end{aligned} \tag{3}$$

where $c = 1$ is the speed of the wave.

4.2. Wave equation 2D

We extend the wave equation in 2D with periodic BCs to explore the requirements for adding the additional spatial dimension. The equation for the evolved field $u(x,y,t)$ is given by

$$\begin{aligned} &u_{tt} + c^2\left(u_{xx} + u_{yy}\right) = 0 \\ &u\left(x,y,0\right) = u_0\left(x,y\right), \\ &x,y \in [0,1),\ t \in [0,1], \end{aligned} \tag{4}$$

where as before the speed of the wave is set to $c = 1$.

4.3. Wave equation 2D non-constant coefficients

By adding a spatially variable wave speed, we study the performance of PINOs in problems with non-constant coefficients. This variable wave speed $c(x,y)$ was incorporated into the PINO by treating it as another randomly generated input field. To produce smoother training data, we set the spatial scale $l = 0.5$ parameter for the wave speed input field $c(x,y)$, though we still used $l = 0.1$ for the initial data. The equation for the evolved field $u(x,y,t)$ is now given by

$$\begin{aligned} &u_{tt} + c(x,y)^2\left(u_{xx} + u_{yy}\right) = 0 \\ &u\left(x,y,0\right) = u_0\left(x,y\right), \\ &x,y \in [0,1),\ t \in [0,1]. \end{aligned} \tag{5}$$

4.4. Burgers equation 1D

The 1D Burgers equation with periodic BCs serves as a nonlinear test case with for a variety of numerical methods. This allowed us to verify that our PINOs can learn and reconstruct nonlinear phenomena. The equation for the field $u(x,y,t)$ is given in conservative form by

$$\begin{aligned} u_t + \partial_x\left(u^2/2\right) &= \nu u_{xx}, \\ u(x,0) &= u_0(x), \\ x \in [0,1),\, t &\in [0,1], \end{aligned} \tag{6}$$

where the viscosity $\nu = 0.01$.

4.5. Burgers equation 2D scalar

To verify our model can handle nonlinear phenomena in 2D, we extend the Burgers equation into 2D by assuming the field $u(x,y,t)$ is a scalar. The equations take the form

$$\begin{aligned} u_t + \partial_x\left(u^2/2\right) + \partial_y\left(u^2/2\right) &= \nu\left(u_{xx} + u_{yy}\right), \\ u(x,y,0) &= u_0(x,y), \\ x,y \in [0,1),\, t &\in [0,1], \end{aligned} \tag{7}$$

where the viscosity $\nu = 0.01$.

4.6. Burgers equation 2D inviscid

We also looked at cases involving the inviscid Burgers equation in 2D in which we set the viscosity $\nu = 0$. This setup is known to produce shocks that can result in numerical instabilities if not handled correctly. We used a FVM to generate this data to ensure stability in the presence of shocks. In turn, this allowed us to investigate the network's performance when processing shocks. The equations are given by

$$\begin{aligned} u_t + \partial_x\left(u^2/2\right) + \partial_y\left(u^2/2\right) &= 0, \\ x,y \in [0,1),\, t &\in [0,1], \\ u(x,y,0) &= u_0(x,y). \end{aligned} \tag{8}$$

We observed that the presence of the shock prevented our Fourier derivative method from producing accurate residuals when we included the full equation in our physics loss term. Instead, for our physics loss, we used the conserved quantity,

$$\int_\Omega u(x,y,t)\,\mathrm{d}x\,\mathrm{d}y = \int_\Omega u(x,y,0)\,\mathrm{d}x\,\mathrm{d}y = C, \tag{9}$$

where C is a constant and Ω is the domain. In other words, we ensured that the total u at every time instance is equal to the total u at $t = 0$.

4.7. Burgers equation 2D vector

We then looked at a vectorized form of the 2D Burgers equation with periodic BCs. This allowed us to test how well the model handles the coupled fields $u(x,y,t)$ and $v(x,y,t)$ that parameterize the system. The equations take the form

$$\begin{gathered} u_t + uu_x + vu_y = \nu\left(u_{xx} + u_{yy}\right), \\ v_t + uv_x + vv_y = \nu\left(v_{xx} + v_{yy}\right), \\ u(x,y,0) = u_0(x,y),\, v(x,y,0) = v_0(x,y), \\ x,y \in [0,1),\, t \in [0,1] \end{gathered} \tag{10}$$

where the viscosity $\nu = 0.01$. We note that this system of equations does not have a conservative form as there is not a continuity equation in this system.

Although coupled equations might seem like a trivial case, there are actually a number of complexities that arise when changing the number of fields. First, having multiple inputs and outputs results in an expanded parameter space for the PINO. Thus, one would expect the network to require a larger volume of training data to produce accurate results. Moreover, PINOs must solve multiple equations simultaneously in the coupled case. In turn, the models must not only solve for single fields, but also compute the contribution of those fields on the other fields. Therefore, it is important to understand how PINOs can resolve coupled fields in this relatively simple 2D vectorized Burgers equation.

4.8. Linear shallow water equations 2D

To examine the properties of PINOs with three coupled equations, we examined the ability of the networks to reproduce the linear shallow water equations with periodic BCs. We assumed that the height of the perturbed surface $h(x,y,t)$ is initially perturbed, but the initial velocity fields $u(x,y,t)$ and $v(x,y,t)$ are initially zero. These equations can be expressed as

$$\frac{\partial h}{\partial t} + H\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0, \tag{11}$$

$$\frac{\partial u}{\partial t} - fv = -g\frac{\partial h}{\partial x}, \tag{12}$$

$$\frac{\partial v}{\partial t} + fu = -g\frac{\partial h}{\partial y}, \tag{13}$$

with $h(x,y,0) = h_0(x,y),\ u(x,y,0) = 0,\ v(x,y,0) = 0,\ x,y \in [0,1),\ t \in [0,1]$, where the gravitational constant $g = 1$, the mean fluid height $H = 100$, and we considered two cases for the Coriolis coefficient $f = \{0,1\}$.

The challenge in this case is that we have three coupled fields with one of them, the perturbed surface height h having a very different physical meaning than the others. Moreover, h is typically of much larger magnitude than either of the velocity fields. By simplifying to a linear problem, we assess the ability of PINOs to reproduce the results of magnitude varying fields without complicated nonlinear terms. We note that coupled equations with terms of varying magnitude are known to be difficult for traditional PINNs as one needs to carefully weight and normalize the equations.

4.9. Nonlinear shallow water equations 2D

Finally, we examined the network performance on the nonlinear shallow water equations. We assumed a similar setup as in the linear case where we assumed the total fluid column height $\eta(x,y,t)$, was given by a mean value of 1 plus some initial perturbation. We again assumed the initial velocity fields $u(x,y,t)$ and $v(x,y,t)$ were zero. These equations are given by

$$\frac{\partial(\eta)}{\partial t} + \frac{\partial(\eta u)}{\partial x} + \frac{\partial(\eta v)}{\partial y} = 0, \tag{14}$$

$$\frac{\partial(\eta u)}{\partial t} + \frac{\partial}{\partial x}\left(\eta u^2 + \frac{1}{2}g\eta^2\right) + \frac{\partial(\eta uv)}{\partial y} = \nu\left(u_{xx} + u_{yy}\right), \tag{15}$$

$$\frac{\partial(\eta v)}{\partial t} + \frac{\partial(\eta uv)}{\partial x} + \frac{\partial}{\partial y}\left(\eta v^2 + \frac{1}{2}g\eta^2\right) = \nu\left(v_{xx} + v_{yy}\right), \tag{16}$$

with $\eta(x,y,0) = \eta_0(x,y),\ u(x,y,0) = 0,\ v(x,y,0) = 0,\ x,y \in [0,1),\ t \in [0,1]$, where the gravitational coefficient $g = 1$ and the viscosity coefficient $\nu = 0.002$ to prevent the formation of shocks.

Figure 2. The training loss curves for the first 150 epochs of training for the nonlinear shallow water equations. Here we compare the different losses from the data, PDE, and ICs as well as their weighted total training loss.

Table 1. Summary of PINO results The first column describes the modeled equation. The second and third columns describe the spatial and temporal resolution, respectively. The fifth and sixth columns display the number of training and testing samples used. The final column provides the relative mean squared error (MSE) of our physics informed neural operators on the test set. To showcase the enhanced performance of PINOs to solve these equations, we also present FNO results for five different equations.

Model	Spatial resolution	Time steps	Training samples	Testing samples	Relative MSE
Wave equation 1D	128	101	900	100	1.22×10^{-03}
Wave equation 2D	128×128	101	45	25	6.60×10^{-03}
Wave equation 2D non-constant coefficients	128×128	101	175	25	4.86×10^{-02}
Burgers equation 1D	128	101	90	10	1.25×10^{-03}
Burgers equation 1D FNO	128	101	90	10	3.86×10^{-03}
Burgers equation 2D scalar	128×128	101	45	25	3.56×10^{-03}
Burgers equation 2D scalar FNO	128×128	101	45	25	6.29×10^{-03}
Burgers equation 2D inviscid	128×128	101	90	10	3.56×10^{-02}
Burgers equation 2D vector	128×128	101	475	25	8.49×10^{-03}
Linear shallow water equations 2D $f = 0$	128×128	101	45	25	6.86×10^{-03}
Linear shallow water equations 2D $f = 0$ FNO	128×128	101	45	25	8.13×10^{-03}
Linear shallow water equations 2D $f = 1$	128×128	101	45	25	6.28×10^{-03}
Linear shallow water equations 2D $f = 1$ FNO	128×128	101	45	25	6.52×10^{-03}
Nonlinear shallow water equations 2D	128×128	101	45	25	1.50×10^{-02}
Nonlinear shallow water equations 2D FNO	128×128	101	45	25	4.40×10^{-02}

This case combines the difficulty of having three coupled fields with complicated nonlinear governing equations. This case provides a very useful and physically interesting benchmark for PINOs as these equations model tsunamis. Moreover, these equations take a similar form to the equations of compressible flow, but without an additional equation for energy that is dependent on the equation of state.

All these different PDEs serve the purpose of establishing the accuracy and reliability of our PINOs, and then explore their application for more interesting scenarios for the 2D Burgers equation, and 2D linear and nonlinear shallow waters equations, which involve several coupled equations.

5. Results

We now quantify the ability of our PINOs to learn the physics described by the PDEs described above. In figures 3–12 we present qualitative and quantitative results that illustrate the performance of our PINOs. We use two types of quantitative metrics, namely, absolute error (shown in each figure) and MSE (summarized in table 1) between PINO predictions and ground truth simulations. While the figures below provide snapshots of the performance of our PINOs at a given time, t, we also provide interactive visualizations of these results at this website. We also provide a tutorial to use our PINOs and reproduce our results in the Data and Learning Hub for Science [42, 43].

Before discussing specific results, we also present the training curves of the nonlinear shallow water equations over its first 150 epochs in figure 2.

In terms of benchmarks, and assuming a single NVIDIA A100 GPU in the ThetaGPU supercomputer at the Argonne Leadership Supercomputing Facility (ThetaGPU is an NVIDIA DGX A100-based system. The DGX A100 comprises eight NVIDIA A100 GPUs that provide a total of 320 GB of memory for training AI datasets, as well as high-speed NVIDIA Mellanox ConnectX-6 network interfaces), we found that:

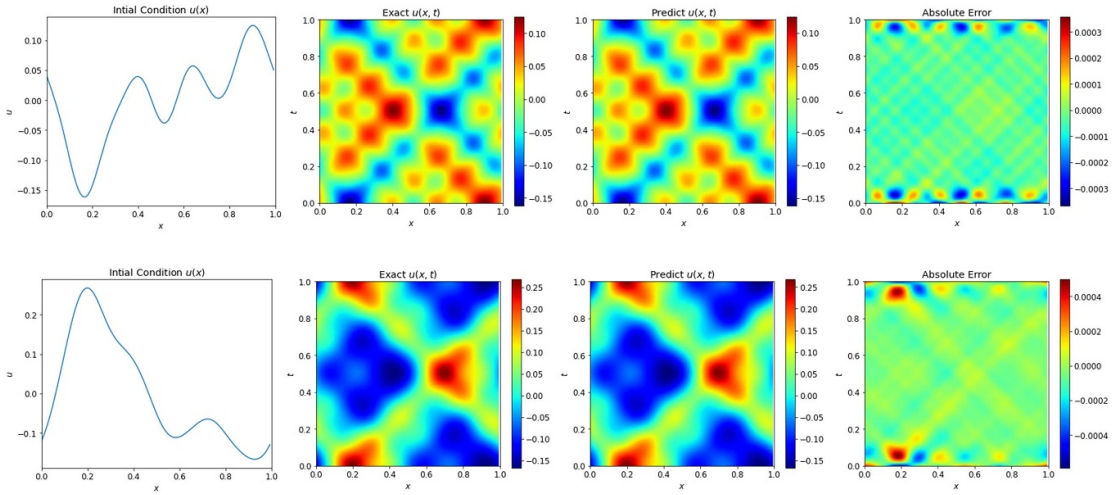

Figure 3. PINO for 1D wave equation test set initial conditions (left column) fed into our neural networks. The center left column displays the ground truth value of $u_{\text{true}}(x, t)$ as produced by our simulations. The center right column shows the value of $u_{\text{pred}}(x, t)$ predicted by our PINO. The right column illustrates the error between the PINO predictions and the ground truth, $u_{\text{pred}} - u_{\text{true}}$.

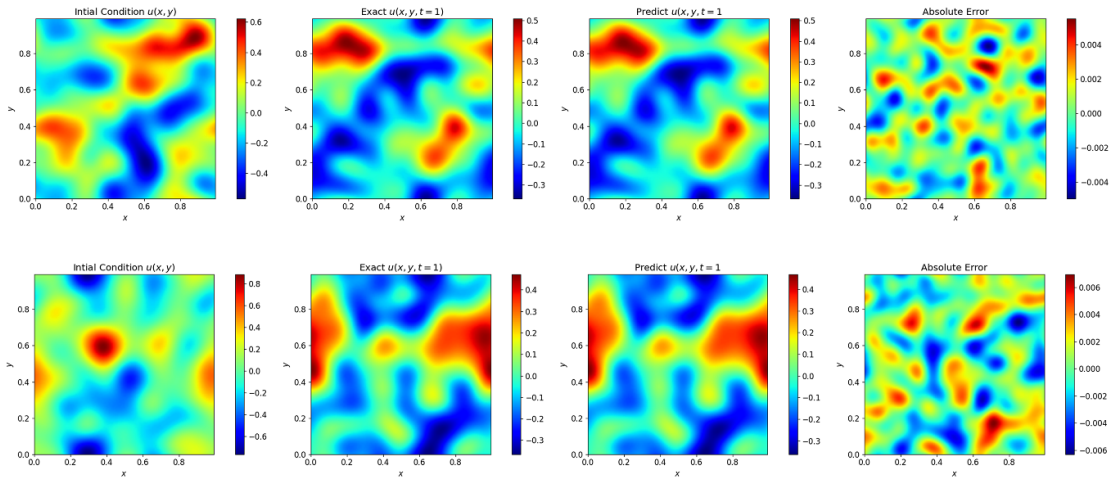

Figure 4. PINO for 2D wave equation. As figure 3, but now in 2D. The left column represents the test set initial conditions that we feed into our PINOs. We evolved the systems until $t = 1$ and present ground truth solutions (center left) and PINO predictions (center right). Even after evolving these simulations until the end of the time domain under consideration, our PINOs predict with excellent accuracy the evolution of the system, as shown in the right column which shown at $t = 1$, $u_{\text{pred}} - u_{\text{true}}$.

- training the 1D Burgers takes 55 s (500 epochs) and a single run takes 0.1 s
- training the 2D linear shallow water equations takes 26 min (150 epochs) and a single run takes 1.79 s
- training the 2D nonlinear shallow water equations takes 26 min (150 epochs) and a single run takes 2.45 s.

5.1. Wave equation 1D results

Figure 3 shows that our PINO for the 1D wave equation learns and describes the physics of this PDE with remarkable accuracy. These results serve the purpose of validating our methods with a simple PDE. We found that the MSE in this case was 1.22×10^{-3} on the test dataset. We note that for this case, we experimented with using a very high resolution of 4096 grid points to generate the training data, but downsampled by a factor of 32 to a final resolution of 128 grid points that were fed into the network. This simulates having available less data to reproduce a result than was required to compute it.

5.2. Wave equation 2D results

Next we consider the 2D wave equation. Figure 4 summarizes our results. As in the 1D scenario, our PINOs have learned the physics of the 2D wave equations with excellent accuracy. Specifically, we found the MSE on the test dataset for this case to be 6.70×10^{-3}.

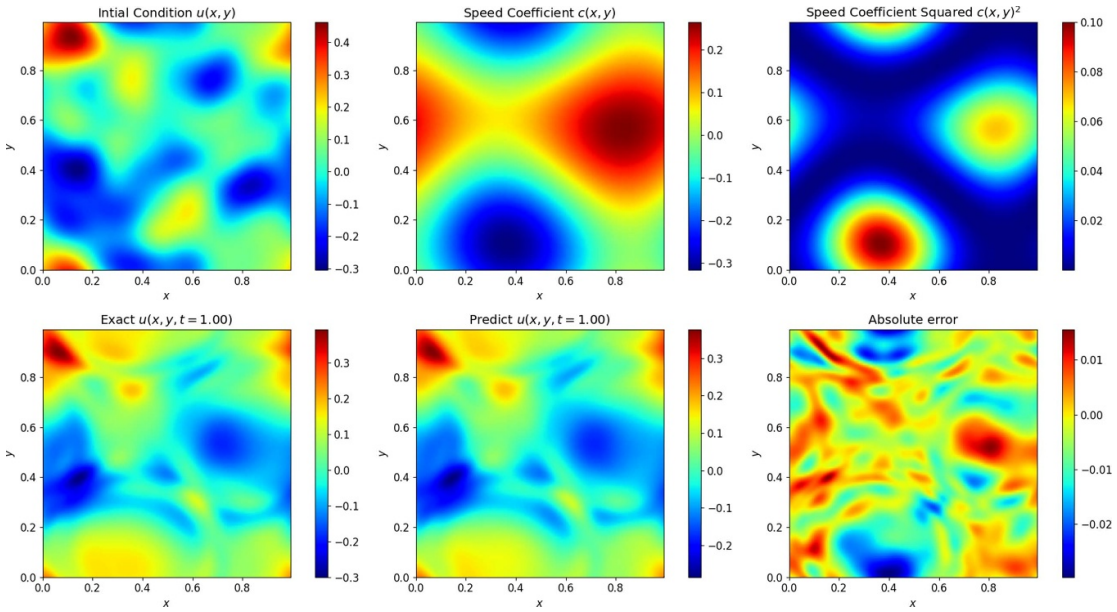

Figure 5. PINO for 2D non-constant coefficients wave equation Similar to figure 4, but here we have a non-constant wave speed. The top row depicts the input fields into the ANN, which are the initial condition u, the wave speed coefficient c, and the wave speed coefficient squared c^2 from left to right respectively. As in the constant coefficient case, we evolved the systems until $t = 1$. On the bottom row, we present the values at $t = 1$ for the ground truth solutions (left), the PINO predictions (center), the error $u_{\text{pred}} - u_{\text{true}}$.

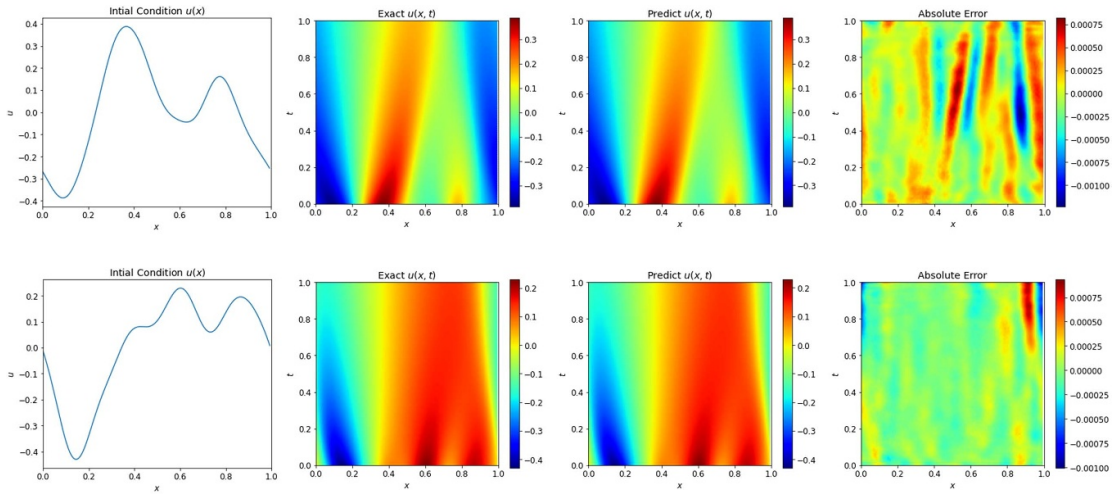

Figure 6. PINO for 1D Burgers equation. The left column shows the test set initial conditions fed into our PINOs. The center left column shows the ground truth value, $u_{\text{true}}(x,t)$, produced by our simulations. The center right column presents the predicted values $u_{\text{pred}}(x,t)$ by our PINOs. The right column shows the discrepancy between PINO and ground truth predictions, $u_{\text{pred}} - u_{\text{true}}$.

5.3. Wave equation 2D non-constant coefficients results

Next we consider the variation of the 2D wave equation with non-constant coefficients. Figure 5 illustrates the results for this problem. As this scenario is considerably more difficult than the uniform coefficient wave equation, the PINO performance is slightly worse, with an MSE of on the test dataset of 0.0486.

5.4. Burgers equation 1D results

We now turn our attention to the Burgers equation, and begin this analysis with a simple and illustrative case, namely the 1D Burgers equation, shown in figure 6. These results show that our PINOs have accurately learned the physics described by this PDE with an excellent level of accuracy. Quantitatively, the MSE on the test dataset was 1.25×10^{-3}. These results furnish evidence that our methods can reproduce results published elsewhere in the literature [27].

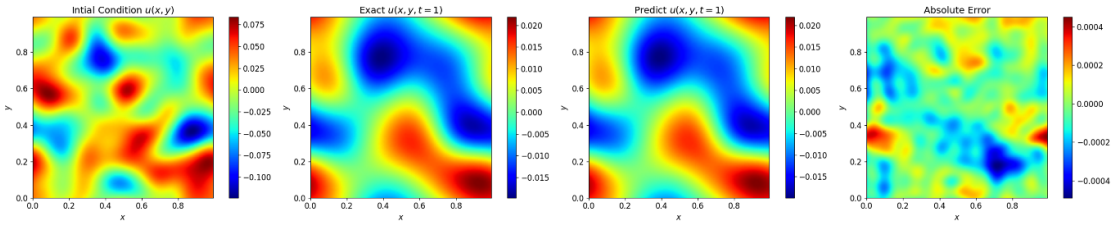

Figure 7. PINO for 2D scalar Burgers equation. Displays the results of the PINO model for this equation. The left column shows a sample of test set initial conditions fed into our models. The center left column displays the ground truth value, $u_{\text{true}}(x, y)$, of our simulations once they have evolved up to $t = 1$. The center right column shows model predictions for this PDE at $t = 1$. We have selected this time to quantify the accuracy of our model once the system has evolved sufficient time to accumulate errors. The right column illustrates the discrepancy between model predictions and the ground truth, $u_{\text{pred}} - u_{\text{true}}$, at $t = 1$.

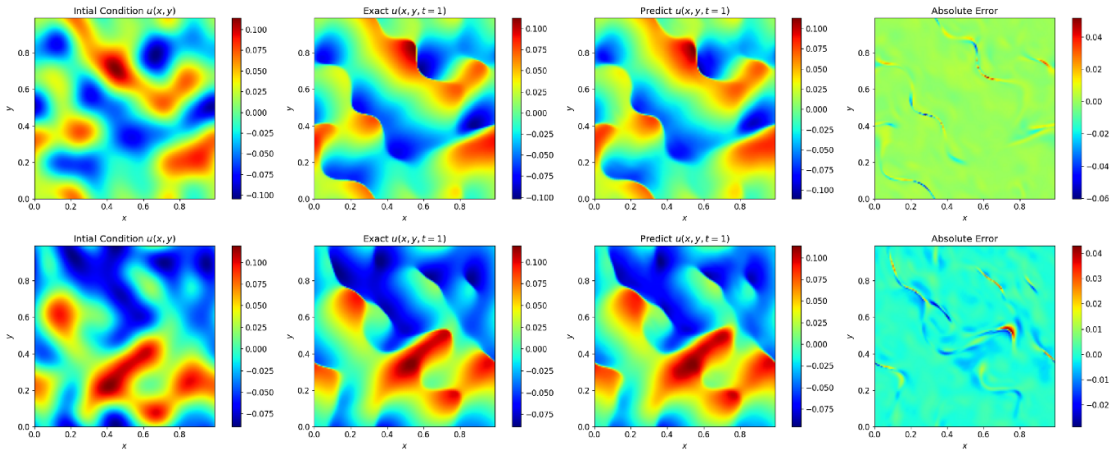

Figure 8. PINO for 2D inviscid Burgers equation. As figure 7 but now in the inviscid case given by equation (8).

5.5. Burgers equation 2D scalar results

This PDE is given by equation (7). As shown in figure 7 our PINOs can learn and describe the physics of this PDE accurately. Note that we present results for this PDE once the system has been evolved throughout the time domain under consideration, i.e. $t \in [0, 1]$. By presenting results at $t = 1$ we gain a good understanding of the actual performance of our PINOs once they have evolved in time and accumulated numerical errors that may depart from ground truth values. For this case, we compared the results to a FNO which was trained with the same architecture, but without a physics informed loss component. In figure 7 we present the results of the PINO and the FNO for the same IC. We found that the MSE for the full test dataset was 3.56×10^{-3} for the PINO and was 6.29×10^{-3} for the FNO.

5.6. Burgers equation 2D inviscid results

This PDE is given by equation (8). As we mentioned before, shocks are a common occurrence for this PDEs, and may lead to numerical instabilities if we do not use shock capturing schemes. We have quantified the ability of our PINOs to handle shocks. We found the PINO to have an MSE of 0.0356 on the test dataset. In figure 8 we notice that our PINOs are able to describe the physics of this PDE with excellent accuracy. However, we observe a significant discrepancy between ground truth and AI predictions right at the regions where shocks occur. These findings indicate that further work is needed to better handle PDEs that involve shocks. This is a specific area of work that we will pursue in the future.

Specifically, such work would look at improvements to the way the model and loss function handle discontinuities in the data. For example, Fourier transforms, which are employed in the neural network and to represent derivatives in the loss function, are known to be highly sensitive to discontinuities. Thus, potential improvements may encompass the use of Galerkin neural networks [44] to better handle discontinuities, and loss functions that incorporate particle number conservation. These improvements should be considered in future work.

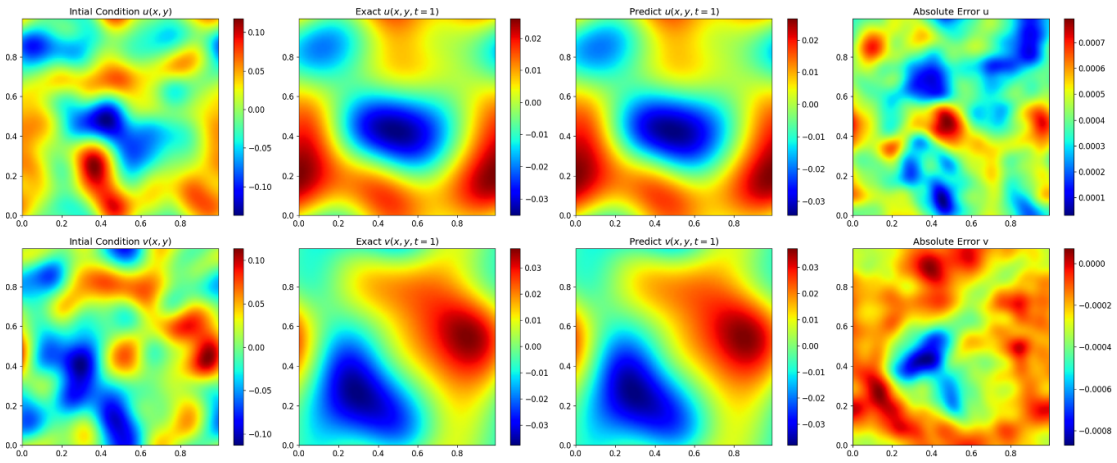

Figure 9. PINO for 2D vector Burgers equation. The top and bottom rows show the u and v fields, respectively. The left column shows test set initial conditions for (u, v). The center left column shows ground truth values for the (u, v) fields once this PDE has been evolved up to $t = 1$. The center right column shows PINO results for the (u, v) fields at $t = 1$. The right column show the discrepancy between PINO and ground truth values at $t = 1$.

5.7. Burgers equation 2D vector results

This is the most complex PDE of the Burgers family we consider in this study, see equation (10). The novel feature of this PDE is that we now need to treat two different fields (u, v). In figure 9 we present a sample result from the test dataset, which demonstrate that our PINOs have learned the physics of this PDE and describe it with remarkable precision even after we have evolved this system until the end of the time domain under consideration. Quantitatively, we achieved an MSE of 8.49×10^{-3} on the test dataset.

These results complete our analysis for a variety of PDEs that involve the Burgers equation, and indicate that for various levels of complexity and ICs our PINOs are capable of learning and describing the physics of these PDEs with excellent accuracy. We have also realized that we need to further develop these methods for PDEs that involve shocks. We are keenly interested in this particular case, and will explore in future work.

5.8. Linear shallow water equations 2D results

Another original result in this study is the use of PINOs to learn the physics of three coupled PDE equations. The first case under consideration is the 2D linear shallow water equation given by equations (11)–(13). In this case we now consider the height of the perturbed surface, h, and the fields (u, v). Figures 10 and 11 present results assuming two Coriolis coefficient $f = \{0, 1\}$, respectively. The discrepancy between PINO predictions and ground truth values in the figures is very small, which furnishes strong evidence for the adequacy of PINOs to learn the physics of this PDE Highlights of these results include:

- **f = 0 case:** Using 45 training samples and 25 testing samples, the MSE on the test dataset was 1.25×10^{-3}.
- **f = 1 case:** With the same number of samples as the previous case, the MSE on the test dataset was calculated to be 6.28×10^{-3}.

These results provide a glimpse of the capabilities of PINOs to learn the physics of these linear, coupled PDEs. We have extensively tested these equations and found that they are robust to a broad range of initial data. These results provided enough motivation to explore the use of PINOs for a more challenging set of coupled PDEs, namely, the 2D nonlinear shallow water equations that we discuss next.

5.9. Nonlinear shallow water equations 2D results

The final original contribution of this study is the use of PINOs to learn the physics of the 2D nonlinear shallow water equation, given by equations (14)–(16). Even though this PDE is significantly more complex than its linear counterpart, we notice in figure 12 that our PINOs learn the physics described by the fields (η, u, v) with remarkable accuracy. We found the MSE for the test dataset of this case to be 0.0150

Finally, we provide an additional metric to quantify the overall performance of all the PDEs we have used in this study in table 1. These results indicate that our PINOs provide state-of-the-art results to model simple PDEs (1D wave equation and 1D Burgers equations), and excellent performance for a variety of PDEs that are solved for the first time in the literature with PINOs.

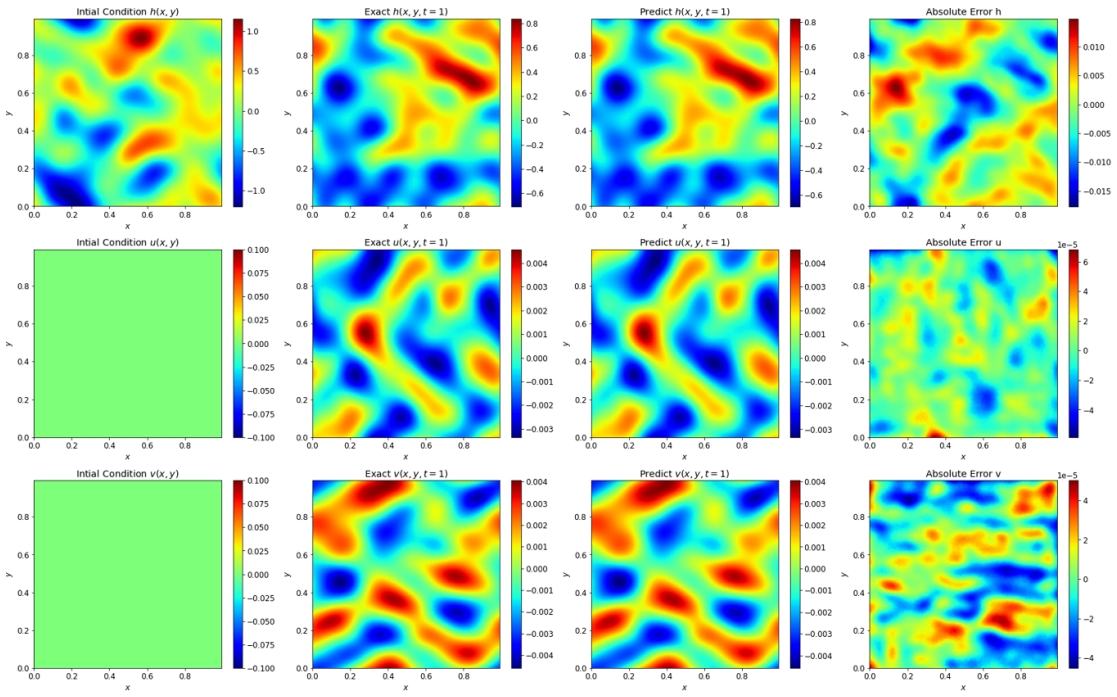

Figure 10. PINO for 2D linear shallow water equation. Assuming a system with a Coriolis coefficient $f = 0$, we show the h, u, and v fields in the top, middle, and bottom rows, respectively. The left column shows the initial condition provided to the network. The center left column displays the ground truth value at $t = 1$ as produced by the simulation. The center right column shows the value of prediction at $t = 1$ predicted by our PINO. The right column indicates the discrepancy between PINO predictions and the ground truth at $t = 1$.

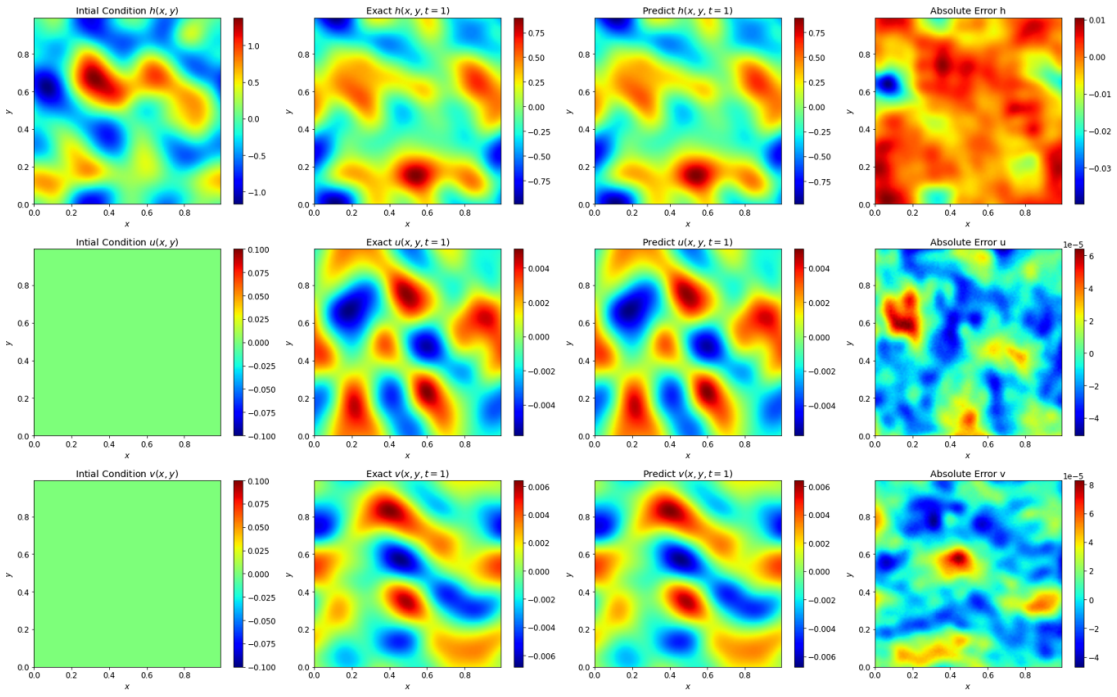

Figure 11. PINO for 2D linear shallow water equation. As figure 10 but now with a Coriolis coefficient $f = 1$.

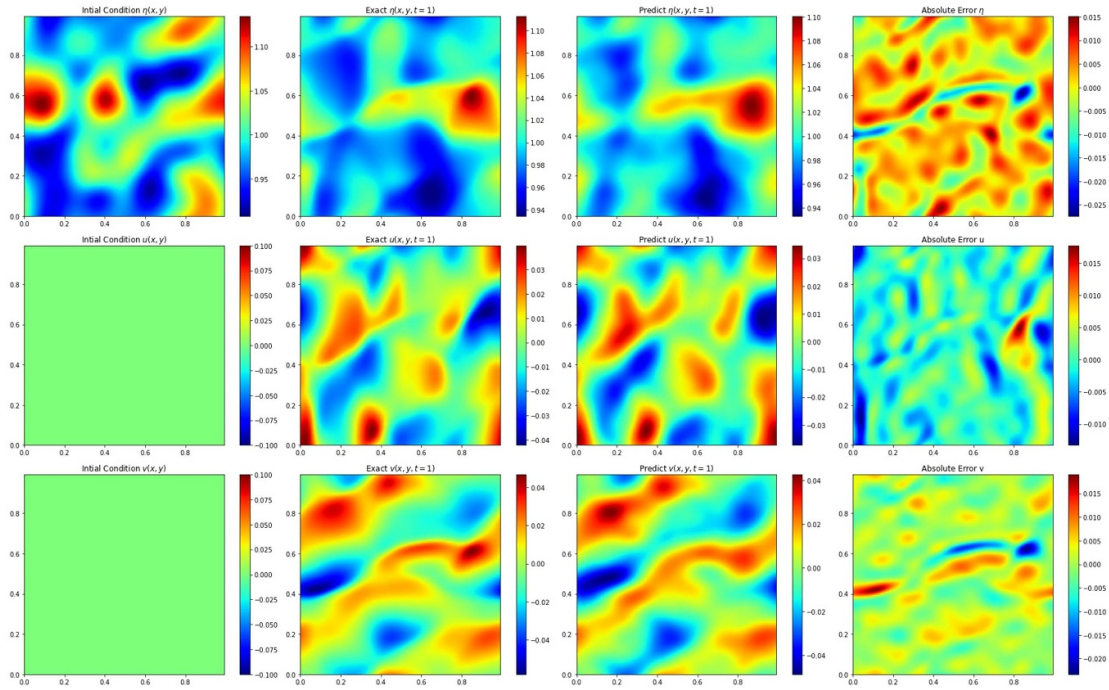

Figure 12. PINO for 2D nonlinear shallow water equation The (η, u, v) fields are shown in the top, middle, and bottom rows, respectively. The left column presents the test set initial condition that is fed into our network. The center left column displays the ground truth value at $t = 1$ as produced by our simulations. The center right column shows the value of prediction at $t = 1$ predicted by the PINO. The right column illustrates the discrepancy between the PINO predictions and the ground truth at $t = 1$.

6. Conclusions

We have introduced an end-to-end framework to learn PDEs that range from simple equations that serve the purpose of testing our methods (1D wave equation and 1D Burgers equation) to increasingly more complex equations (2D scalar, 2D inviscid and 2D vector Burgers equation), and coupled PDEs that are solved with PINOs for the first time in the literature (2D linear and nonlinear shallow waters equations). The methods we introduce in this study provide the flexibility to produce initial data to test the robustness and applicability of AI surrogates for a broad range of physically motivated scenarios. We provide scientific visualizations of our results through an interactive website. We also release our PINOs and scientific software through the Data and Learning Hub for Science so that AI practitioners may download, use and further develop our neural networks. In addition to this Data and Learning Hub for Science implementation, we release the source code used in this paper.

Future work may focus on the extension of these PINOs to high-dimensional PDEs that are relevant for the modeling of complex phenomena that demand subgrid scale precision, and which typically lead to computationally expensive simulations. We will also focus on developing methods that handle shocks effectively, since these phenomena are common in fluid dynamics and relativistic astrophysics, to mention a few. We will also continue our research program combining scientific visualization and accelerated computing to gain insights about what PINOs learn from data, and how this information may be used to enhance their performance when applied to experimental datasets [45–47].

It is our expectation that our AI surrogates may be used to replace computationally demanding numerical methods to learn PDEs in scientific software used to model multi-scale and multi-physics phenomena—e.g. numerical relativity simulations of gravitational wave sources, weather forecasting, etc—and eventually provide data-driven and physics informed solutions that more accurately describe and identify novel features and patterns in experimental data.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/shawnrosofsky/PINO_Applications.

Acknowledgments

This material is based upon work supported by Laboratory Directed Research and Development (LDRD) funding from Argonne National Laboratory, provided by the Director, Office of Science, of the U S Department of Energy under Contract No. DE-AC02-06CH11357. This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. S R and E A H gratefully acknowledge National Science Foundation Award OAC-1931561. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation Grant Number ACI-1548562. This work used the Extreme Science and Engineering Discovery Environment (XSEDE) Bridges-2 at the Pittsburgh Supercomputing Center through allocation TG-PHY160053. This research used the Delta advanced computing and data resource which is supported by the National Science Foundation (Award OAC-2005572) and the State of Illinois. Delta is a joint effort of the University of Illinois Urbana-Champaign and its National Center for Supercomputing Applications.

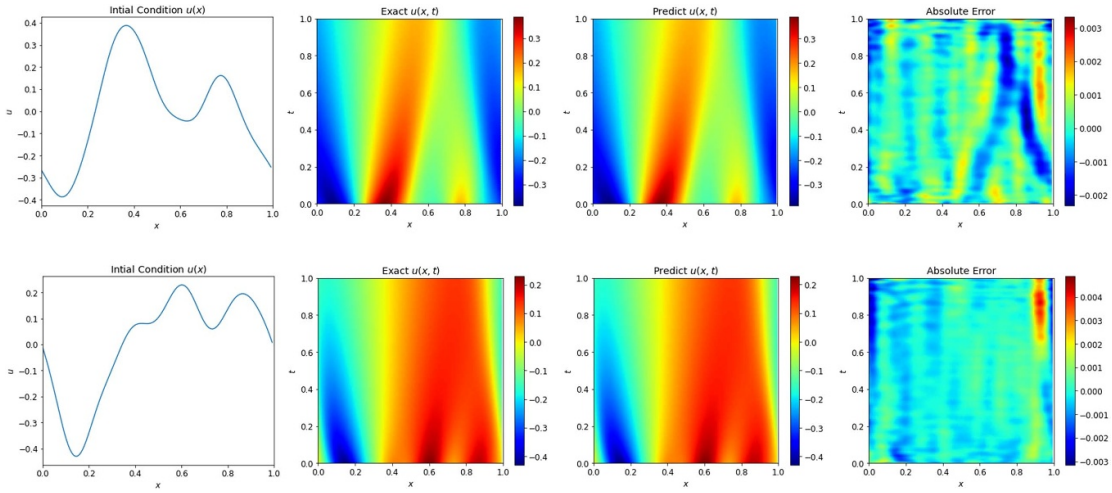

Figure 13. FNO for 1D Burgers equation. The left column shows the test set initial conditions fed into our FNO. The center left column shows the ground truth value, $u_{\text{true}}(x,t)$, produced by our simulations. The center right column presents the predicted values $u_{\text{pred}}(x,t)$ by our FNO. The right column shows the discrepancy between FNO and ground truth predictions, $u_{\text{pred}} - u_{\text{true}}$.

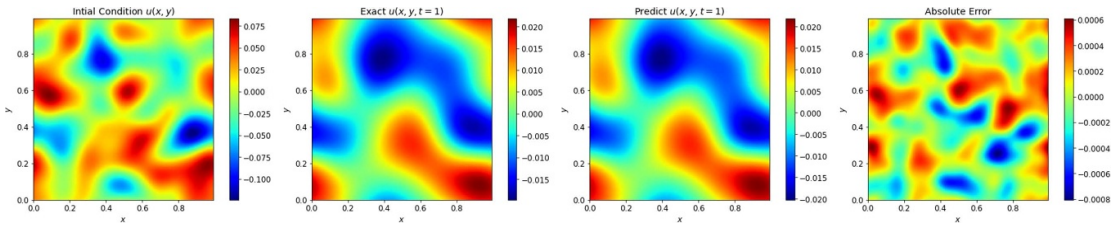

Figure 14. FNO for 2D scalar Burgers equation The left column shows a sample of test set initial conditions fed into our FNO model. The center left column displays the ground truth value, $u_{true}(x,y)$, of our simulations once they have evolved up to $t = 1$. The center right column shows model predictions for this PDE at $t = 1$. We have selected this time to quantify the accuracy of our model once the system has evolved sufficient time to accumulate errors. The right column illustrates the discrepancy between model predictions and the ground truth, $u_{\text{pred}} - u_{\text{true}}$, at $t = 1$.

Appendix. Fourier neural operators

Here we present results for the solution of three different types of PDEs using Fourier neural operators (FNOs). Specifically, we present results for the 1D Burgers equation in figure 13; the 2D scalar Burgers equation in figure 14; and the 2D nonlinear shallow water equation in figure 15. These results indicate that including physics principles in the optimization of AI surrogates leads to enhanced performance. We also refer the reader to table 1, where we show additional comparisons between PINOs and FNOs for other PDEs in terms of relative MSE.

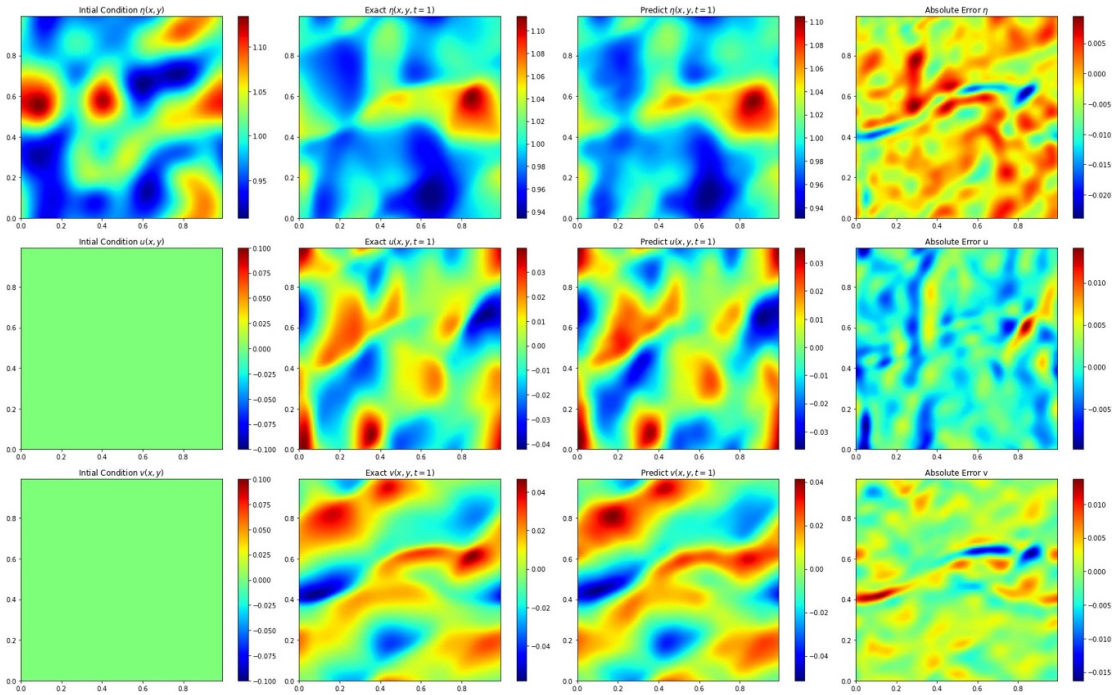

Figure 15. FNO for 2D nonlinear shallow water equation The (η, u, v) fields are shown in the top, middle, and bottom rows, respectively. The left column presents the test set initial condition that is fed into our network. The center left column displays the ground truth value at $t = 1$ as produced by our simulations. The center right column shows the value of prediction at $t = 1$ predicted by the FNO. The right column illustrates the discrepancy between the FNO predictions and the ground truth at $t = 1$.

ORCID iDs

Shawn G Rosofsky https://orcid.org/0000-0002-3319-576X
E A Huerta https://orcid.org/0000-0002-9682-3604

References

[1] Geroch R P 1996 *46th Scottish Universities Summer School in Physics: General Relativity*
[2] Press W H, Teukolsky S A, Vetterling W T and Flannery B P 2007 *Numerical Recipes 3rd Edition: The Art of Scientific Computing* 3rd edn (Cambridge: Cambridge University Press)
[3] Radice D 2020 *Symmetry* **12** 1249
[4] Radice D, Bernuzzi S, Perego A and Haas R 2021 arXiv:2111.14858 [astro-ph.HE]
[5] Foucart F 2020 *Front. Astron. Space Sci.* **7** 46
[6] Schalkwijk J, Jonker H J J, Siebesma A P and Meijgaard E V 2015 *Bull. Am. Meteorol. Soc.* **96** 715
[7] Erba A, Baima J, Bush I, Orlando R and Dovesi R 2017 *J. Chem. Theory Comput.* **13** 5019
[8] Asch M *et al* 2018 *Int. J. High Perform. Comput. Appl.* **32** 435
[9] Gropp W, Banerjee S and Foster I 2020 arXiv:2012.09303 [cs.CY]
[10] Huerta E A 2020 *J. Big Data* **7** 88
[11] Taher M 2009 *2009 4th Int. Design and Test Workshop* (*IDT*) pp 1–6
[12] Rodrigues C I, Hardy D J, Stone J E, Schulten K and Hwu W-M W 2008 *Proc. 5th Conf. on Computing Frontiers CF '08* (*Association for Computing Machinery* (New York)) pp 273–82
[13] Wysocki D, O'Shaughnessy R, Lange J and Fang Y-L L 2019 *Phys. Rev.* D **99** 084026
[14] Graff P, Feroz F, Hobson M P and Lasenby A 2012 *Mon. Not. R. Astron. Soc.* **421** 169
[15] Rosofsky S G and Huerta E A 2020 *Phys. Rev.* D **101** 084024
[16] Huerta E A and Zhao Z 2020 Advances in machine and deep learning for modeling and real-time detection of multi-messenger sources *Handbook of Gravitational Wave Astronomy*, ed C Bambi, S Katsanevas and K D Kokkotas (Singapore: Springer) pp 1–27
[17] Cuoco E *et al* 2021 *Mach. Learn. Sci. Technol.* **2** 011002
[18] Huerta E A *et al* 2021 *Nat. Astron.* **5** 1062
[19] Huerta E A *et al* 2019 *Nat. Rev. Phys.* **1** 600
[20] Khan A, Huerta E A and Zheng H 2022 *Phys. Rev.* D **105** 024024
[21] Chaturvedi P, Khan A, Tian M, Huerta E A and Zheng H 2022 *Front. Artif. Intell.* **5** 828672
[22] MacLeod B P *et al* 2020 *Sci. Adv.* **6** eaaz8867
[23] Wilkinson M *et al* 2016 *Sci. Data* **3** 160018
[24] Park H, Zhu R, Huerta E A, Chaudhuri S, Tajkhorshid E and Cooper D 2022 arXiv:2212.11317 [cond-mat.mtrl-sci]
[25] Chen Y *et al* 2022 *Sci. Data* **9** 31
[26] Ravi N, Chaturvedi P, Huerta E A, Liu Z, Chard R, Scourtas A, Schmidt K J, Chard K, Blaiszik B and Foster I 2022 *Sci. Data* **9** 657
[27] Li Z, Zheng H, Kovachki N, Jin D, Chen H, Liu B, Azizzadenesheli K and Anandkumar A 2021 arXiv:2111.03794 [cs.LG]

[28] Raissi M, Perdikaris P and Karniadakis G E 2017 Physics informed deep learning (part i): data-driven solutions of nonlinear partial differential equations (arXiv:1711.10561 [cs.AI])
[29] Raissi M, Perdikaris P and Karniadakis G E 2017 Physics informed deep learning (part ii): data-driven discovery of nonlinear partial differential equations (arXiv:1711.10566 [cs.AI])
[30] Raissi M, Perdikaris P and Karniadakis G 2019 *J. Comput. Phys.* **378** 686
[31] Pang G, Lu L and Karniadakis G E 2019 *SIAM J. Sci. Comput.* **41** A2603
[32] Lu L, Meng X, Mao Z and Karniadakis G E 2021 *SIAM Rev.* **63** 208
[33] Kovachki N, Li Z, Liu B, Azizzadenesheli K, Bhattacharya K, Stuart A and Anandkumar A 2021 arXiv:2108.08481 [cs.LG]
[34] Lu L, Jin P, Pang G, Zhang Z and Karniadakis G E 2021 *Nat. Mach. Intell.* **3** 218–29
[35] Wang S, Wang H and Perdikaris P 2021 *Sci. Adv.* **7** eabi8605
[36] Li Z, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A and Anandkumar A 2020 Neural operator: graph kernel network for partial differential equations (arXiv:2003.03485 [cs.LG])
[37] Li Z, Kovachki N B, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A M and Anandkumar A 2020 *CoRR* arXiv:abs/2006.09535
[38] Li Z, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A and Anandkumar A 2020 Fourier neural operator for parametric partial differential equations (arXiv:2010.08895 [cs.LG])
[39] Rosofsky S G and Huerta E A arXiv:2302.08332 [physics.comp-ph]
[40] Paszke A *et al* 2019 *Advances in Neural Information Processing Systems* vol **32**
[41] Hendrycks D and Gimpel K 2016 Gaussian error linear units (gelus) (arXiv:1606.08415 [cs.LG])
[42] Chard R, Li Z, Chard K, Ward L, Babuji Y, Woodard A, Tuecke S, Blaiszik B, Franklin M J and Foster I 2019 *2019 IEEE Int. Parallel and Distributed Processing Symp.* (*IPDPS*)
[43] Blaiszik B, Ward L, Schwarting M, Gaff J, Chard R, Pike D, Chard K and Foster I 2019 *MRS Commun.* **9** 1125
[44] Ainsworth M and Dong J 2021 *SIAM J. Sci. Comput.* **43** A2474
[45] Doshi-Velez F and Kim B 2017 arXiv:1702.08608 [stat.ML]
[46] Safarzadeh M, Khan A, Huerta E A and Wattenberg M 2022 arXiv:2202.07399 [gr-qc]
[47] Carvalho D V, Pereira E M and Cardoso J S 2019 *Electronics* **8** 832