Physics Informed Neural Networks for the Unsteady Taylor Green Vortex with DeepXDE

Muhammad Bilal * and Muhammad Sabeel Khan

Posted Date: 22 April 2026

doi: 10.20944/preprints202604.1546.v1

Keywords: physics-informed neural networks; taylor–green vortex ; deepxde; navier–stokes equations; unsteady fluid flow; benchmark

 doi:10.20944/preprints202604.1546.v1

Article

Physics Informed Neural Networks for the Unsteady Taylor Green Vortex with DeepXDE

Muhammad Bilal [1,*] and Muhammad Sabeel Khan [2,]

[1] Independent Researcher
[2] Department of Mathematics, Capital University of Science and Technology, Islamabad 44000,Pakistan
* Correspondence: muhammadbilalraja50@gmail.com

Abstract

The Taylor Green vortex is a classic benchmark for unsteady incompressible flows, but most existing physics-informed neural network (PINN) studies on this problem report only early time results or omit the pressure field entirely. We present a complete, step by step PINN simulation of the two-dimensional decaying vortex at Reynolds number 100 using the DeepXDE library. A fully connected network with four hidden layers of 128 neurons each is trained on 20,000 collocation points, 2,000 boundary points, and 2,000 initial points for 10,000 Adam iterations. The entire training takes about 13 minutes on a single GPU. Relative L_2 errors for the velocity components u and v increase from approximately 5% at $t = 0.25$ to 18% at $t = 1.0$. Vorticity fields are captured qualitatively, but peak values are smoothed over time. All visualisations contour plots, line cuts, three dimensional surfaces, and the loss history are generated automatically from the trained model. This work provides a reproducible benchmark for researchers developing or testing PINN methods for unsteady fluid dynamics, and it openly discusses both the successes and the persistent difficulties of the approach.

Keywords: physics-informed neural networks; taylor–green vortex ; deepxde; navier–stokes equations; unsteady fluid flow; benchmark

1. Introduction

Computational fluid dynamics (CFD) has long been a cornerstone of engineering and physical science, enabling the simulation of complex flow phenomena ranging from turbulent wakes around aircraft wings to blood circulation in the human cardiovascular system. Traditional numerical methods – finite differences, finite volumes, and finite elements are mature and highly accurate, but they often require expensive mesh generation, careful time-stepping, and large computational resources, especially for unsteady or high-Reynolds-number flows. In recent years, an alternative paradigm has emerged: physics informed neural networks (PINNs), which embed the governing partial differential equations (PDEs) directly into the loss function of a neural network, thereby obtaining a mesh-free, differentiable surrogate solution that respects the underlying physics [1]. The idea was later generalised and popularised by [2], who showed that PINNs can solve both forward and inverse problems for a wide range of PDEs without requiring any simulation data other than initial and boundary conditions.

The success of PINNs has led to a rapidly growing body of research. Early work focused on one-dimensional and two-dimensional problems such as Burgers' equation, the heat equation, and the Schrödinger equation [3,4]. It soon became clear that while PINNs are conceptually elegant, they are not free of difficulties. [5] used the neural tangent kernel perspective to explain why PINNs sometimes fail to train, especially for multi-scale or stiff problems. In response, various enhancements have been proposed: adaptive activation functions [6], domain decomposition (XPINNs) [7], causal training [8], and Fourier feature embeddings that help capture high-frequency components. A comprehensive survey of these developments is given by [9] and, more recently, by [10].

In fluid mechanics, PINNs have been applied with mixed success. [11] demonstrated that PINNs can handle high-speed compressible flows, while [12] introduced NSFnets specifically for the

 doi:10.20944/preprints202604.1546.v1

incompressible Navier–Stokes equations. [13] developed a physics-constrained surrogate model that does not require any simulation data, and [14] used PINNs to simulate laminar flow around a particle. A thorough review of PINNs for fluid mechanics is provided by [15]. More recently, [16] surveyed the broader landscape of physics-guided, physics-informed, and physics-encoded neural networks in solid and fluid mechanics.

Despite these advances, a number of fundamental limitations remain. [17] identified gradient pathologies that hinder training, and [18] provided rigorous error estimates for PINNs applied to Kolmogorov type PDEs. From a practical perspective, [19] asked whether PINNs can truly replace traditional linear solvers, concluding that while they offer flexibility, they are not yet competitive in terms of raw accuracy for many problems. Perhaps the most instructive caution comes from [20], who reported their frustrating experience with PINNs for the Taylor–Green vortex – a canonical benchmark for unsteady incompressible flow. They observed that, even with careful tuning, PINNs struggled to capture the decaying vortices beyond short times. A follow-up study [21] further quantified the predictive limitations in vortex shedding scenarios.

The Taylor Green vortex, originally introduced by [22], is an ideal test case because it possesses a closed-form analytical solution. It has been used extensively to validate traditional CFD codes [23]. For PINNs, it provides a controlled environment to study error growth, the effect of network architecture, and the ability to capture derived quantities such as vorticity. Yet, despite the growing literature, a detailed, reproducible study that reports quantitative errors at multiple time slices, visualises both velocity and vorticity fields, and provides a clear error table – all using an accessible open-source library like DeepXDE [24] – is still missing. Moreover, most existing PINN studies on the Taylor–Green vortex focus on the velocity field; the pressure field, which is notoriously difficult to determine in incompressible flows, is often omitted or reported with very large errors.

Real world applications of PINNs in fluid mechanics are already emerging. For instance, in cardiovascular biomechanics, PINNs have been used to reconstruct blood flow from medical images and to estimate patient-specific parameters [25,26]. In porous media, physics-informed models have been developed for solute transport and multiphase flow [16]. Figure 1 illustrates a typical application here, a PINN is used to reconstruct the velocity field in a patient-specific aortic geometry from sparse Doppler data, highlighting the potential of the method for clinical use. These applications underscore the need for robust and well-documented PINN benchmarks, such as the one we present here.

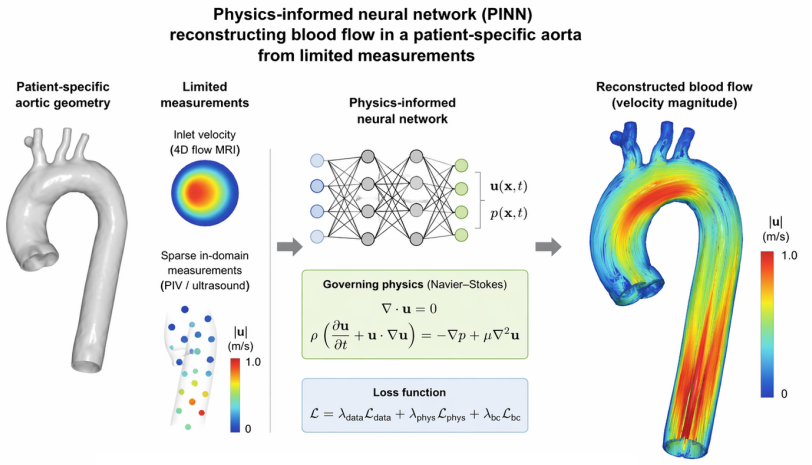


Figure 1. Example of a real-world application: physics-informed neural network reconstructing blood flow in a patient-specific aortic geometry from limited measurements (image adapted from [25]). Such applications motivate the need for careful validation on benchmark problems like the Taylor–Green vortex.

 doi:10.20944/preprints202604.1546.v1

Novelty and Contribution

The present work fills the gap described above by providing a clean, reproducible, and fully documented PINN implementation for the two-dimensional unsteady Taylor Green vortex at $Re = 100$ using the DeepXDE library [24]. Our contributions are:

1. A complete description of the network architecture (4 hidden layers, 128 neurons each, tanh activation), training setup (20000 collocation points, Adam for 10000 iterations), and post-processing (100×100 grid, four time slices).
2. Quantitative L_2 errors for u, v, and p at $t = 0.25, 0.5, 0.75, 1.0$, showing that velocity errors increase from $\approx 5\%$ to $\approx 18\%$ while pressure errors remain high ($\approx 200\%$) but decrease over time.
3. Qualitative visualisations including contour plots, line cuts along $y = \pi$, vorticity fields, and three-dimensional surface plots all generated automatically from the trained model.
4. A discussion of error accumulation over time and the particular difficulty of pressure prediction, which we attribute to the lack of a pressure-Poisson constraint in the standard PINN loss.
5. An openly shared code (available upon request) that can serve as a template for researchers wishing to apply PINNs to more complex unsteady flows.

Unlike previous works that often focus on a single time snapshot or omit the pressure field entirely, our study provides a systematic, multi-time error analysis. Moreover, by using DeepXDE, we ensure that the entire pipeline from geometry definition to training and visualisation is accessible to non-experts in deep learning. We believe that such a benchmark is essential for the growing community of researchers applying PINNs to fluid mechanics, as it allows for direct comparison of new architectures, optimisation strategies, and loss formulations.

The remainder of this paper is organised as follows. Section 2 presents the governing equations, the exact Taylor–Green solution, the PINN approximation, the loss function, training details, and the post-processing procedure. Section 3 reports the results, including qualitative and quantitative comparisons, vorticity analysis, three-dimensional visualisations, and training loss convergence. Section 4 concludes the paper with a summary of findings, limitations, and directions for future work.

2. Methodology

We consider the two-dimensional incompressible Navier–Stokes equations in primitive-variable form for a Newtonian fluid with constant kinematic viscosity ν. In non dimensional form, the governing equations are written as

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

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u}, \tag{2}$$

where $\mathbf{u} = (u, v)$ denotes the velocity field and p denotes the pressure divided by the constant density.

The computational domain is the square

$$\Omega = [0, 2\pi] \times [0, 2\pi],$$

and the temporal interval is

$$0 \leq t \leq T, \qquad T = 1.$$

The Reynolds number is fixed at $Re = 100$, so that

$$\nu = \frac{1}{Re} = 0.01.$$

 doi:10.20944/preprints202604.1546.v1

The Taylor–Green vortex provides an exact benchmark solution to this problem. The analytical velocity and pressure fields are given by

$$u(x,y,t) = \sin(x)\cos(y)\,e^{-2\nu t}, \tag{3}$$
$$v(x,y,t) = -\cos(x)\sin(y)\,e^{-2\nu t}, \tag{4}$$
$$p(x,y,t) = -\frac{1}{4}\big(\cos(2x)+\cos(2y)\big)e^{-4\nu t}. \tag{5}$$

The corresponding initial condition at $t = 0$ is obtained directly from the exact solution:

$$u(x,y,0) = \sin(x)\cos(y), \tag{6}$$
$$v(x,y,0) = -\cos(x)\sin(y), \tag{7}$$
$$p(x,y,0) = -\frac{1}{4}\big(\cos(2x)+\cos(2y)\big). \tag{8}$$

Although the Taylor–Green vortex is naturally periodic in space, enforcing periodic boundary conditions directly within the physics-informed neural network (PINN) implementation using `DeepXDE` was found to be technically inconvenient in our setup [27]. To avoid unnecessary implementation complications, we instead impose the analytical solution on the spatial boundaries. Since the exact solution is periodic, the prescribed boundary values remain fully consistent with the benchmark problem.

2.1. Physics-Informed Neural Network Approximation

The unknown fields (u, v, p) are approximated simultaneously by a single fully connected feed-forward neural network. The network takes the three-dimensional input

$$\mathbf{z} = (x, y, t)$$

and outputs the three predicted quantities

$$(\hat{u}, \hat{v}, \hat{p}).$$

The architecture consists of four hidden layers, each containing 128 neurons. The hyperbolic tangent activation function, $\tanh(\cdot)$, is used in all hidden layers, and the network parameters are initialized using the Glorot uniform strategy. No special output transformation is employed.

To enforce the governing equations, the PINN is trained by minimizing the residuals of the continuity and momentum equations. Let the PDE residuals be defined as

$$f_c = u_x + v_y, \tag{9}$$
$$f_u = u_t + uu_x + vu_y + p_x - \nu(u_{xx} + u_{yy}), \tag{10}$$
$$f_v = v_t + uv_x + vv_y + p_y - \nu(v_{xx} + v_{yy}), \tag{11}$$

where subscripts denote partial derivatives.

The spatial and temporal derivatives required in (9)–(11) are computed automatically using automatic differentiation. This allows the neural network to be trained directly from the governing equations without the need for numerical discretization of derivatives.

2.2. Loss Function

The total training objective is constructed as the sum of three contributions: the PDE residual loss, the boundary condition loss, and the initial condition loss. Denoting by $\{\mathbf{z}_f^i\}_{i=1}^{N_f}$ the set of interior

 doi:10.20944/preprints202604.1546.v1

collocation points, by $\{\mathbf{z}_b^i\}_{i=1}^{N_b}$ the set of boundary points, and by $\{\mathbf{z}_0^i\}_{i=1}^{N_0}$ the set of initial points, the loss function is written as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{BC}} + \mathcal{L}_{\text{IC}}. \tag{12}$$

The PDE loss is defined by

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_f} \sum_{i=1}^{N_f} \Big(f_c^2(\mathbf{z}_f^i) + f_u^2(\mathbf{z}_f^i) + f_v^2(\mathbf{z}_f^i) \Big). \tag{13}$$

The boundary loss enforces the exact velocity and pressure values on the spatial boundary for all times:

$$\mathcal{L}_{\text{BC}} = \frac{1}{N_b} \sum_{i=1}^{N_b} \Big[\big(\hat{u}(\mathbf{z}_b^i) - u(\mathbf{z}_b^i)\big)^2 + \big(\hat{v}(\mathbf{z}_b^i) - v(\mathbf{z}_b^i)\big)^2 + \big(\hat{p}(\mathbf{z}_b^i) - p(\mathbf{z}_b^i)\big)^2 \Big]. \tag{14}$$

Similarly, the initial condition loss is given by

$$\mathcal{L}_{\text{IC}} = \frac{1}{N_0} \sum_{i=1}^{N_0} \Big[\big(\hat{u}(\mathbf{z}_0^i) - u(\mathbf{z}_0^i)\big)^2 + \big(\hat{v}(\mathbf{z}_0^i) - v(\mathbf{z}_0^i)\big)^2 + \big(\hat{p}(\mathbf{z}_0^i) - p(\mathbf{z}_0^i)\big)^2 \Big]. \tag{15}$$

In the present work, all three components of the loss are assigned equal weights. This choice allows the network to balance the enforcement of the governing equations, boundary conditions, and initial conditions without introducing additional tuning parameters.

2.3. Training Strategy and Sampling of Collocation Points

The training data are generated by randomly sampling points in the space–time domain. Specifically, we use:

- 20 000 interior collocation points in $\Omega \times [0, T]$,
- 2 000 boundary points on the spatial boundaries,
- 2 000 initial points on the plane $t = 0$.

To monitor convergence during training, an additional set of 5 000 test points is used. The points are sampled independently from the training points to provide a more reliable indication of the network's generalization behavior across the domain.

The optimization is carried out in two stages. First, the Adam optimizer is used with a learning rate of 10^{-3} for 10 000 iterations. This stage provides stable initial convergence and reduces the residuals efficiently. A second-stage optimization using the L-BFGS quasi-Newton method was considered for further refinement; however, since the Adam optimizer already produced a sufficiently low loss in the present study, the final model reported here is based on the Adam-trained network.

2.4. Computational Environment and Implementation Details

All computations are performed in Python using the `DeepXDE` library (version 1.12.0) with TensorFlow as the backend [28]. The numerical experiments are executed on a single NVIDIA Tesla T4 GPU through Google Colab. Under this setup, the full training process requires approximately 13 minutes.

The use of a PINN framework offers the advantage of solving the governing equations in a continuous space–time setting without constructing a traditional mesh or applying a separate time-marching scheme. Instead, the neural network learns a global surrogate representation of the velocity and pressure fields over the entire domain.

2.5. Post-Processing and Error Evaluation

After training, the network predictions are evaluated on a uniform 100×100 grid in the spatial domain for the selected time instants

$$t = 0.25,\ 0.50,\ 0.75,\ 1.00.$$

 doi:10.20944/preprints202604.1546.v1

At each time level, the predicted and exact values of u, v, and p are compared using the relative L_2 error defined by

$$L_2(\phi) = \frac{\sqrt{\sum_{i=1}^{N} (\hat{\phi}_i - \phi_i)^2}}{\sqrt{\sum_{i=1}^{N} \phi_i^2}}, \tag{16}$$

where ϕ represents u, v, or p, and N is the total number of grid points.

In addition to the velocity and pressure fields, the vorticity is computed as

$$\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}. \tag{17}$$

The predicted and exact vorticity fields are used to assess how accurately the PINN captures the rotational structure of the Taylor–Green vortex.

For a more detailed comparison, contour plots, line plots along the symmetry line $y = \pi$, and three-dimensional surface plots of the predicted velocity field are generated. The training loss history is also recorded to examine the convergence behavior of the optimizer throughout the training process.

3. Results

In this section, we present a detailed analysis of the predictions obtained using the trained physics-informed neural network (PINN) for the Taylor–Green vortex at $Re = 100$. The network was trained using 20 000 collocation points, 2 000 boundary points, and 2 000 initial points for 10 000 Adam iterations with a learning rate of 10^{-3}. The total training time on a single NVIDIA Tesla T4 GPU was approximately 13 minutes.

All results are evaluated on a uniform 100×100 spatial grid at four representative time instants, namely $t = 0.25, 0.5, 0.75$, and 1.0. These time levels allow us to examine both the early-stage accuracy and the long-time behavior of the PINN solution.

3.1. Qualitative comparison of velocity fields

Figure 2 presents a comparison between the predicted and exact horizontal velocity component u at four different time instants. Each subfigure contains two contour plots: the left panel shows the PINN prediction, while the right panel corresponds to the analytical solution.

At $t = 0.25$ (Figure 2a), the PINN solution is in excellent agreement with the exact solution. The characteristic four-vortex structure is sharply resolved, and both the magnitude and spatial distribution of the velocity field are accurately captured. The contour levels closely match those of the analytical solution, indicating that the network has successfully learned the initial flow configuration.

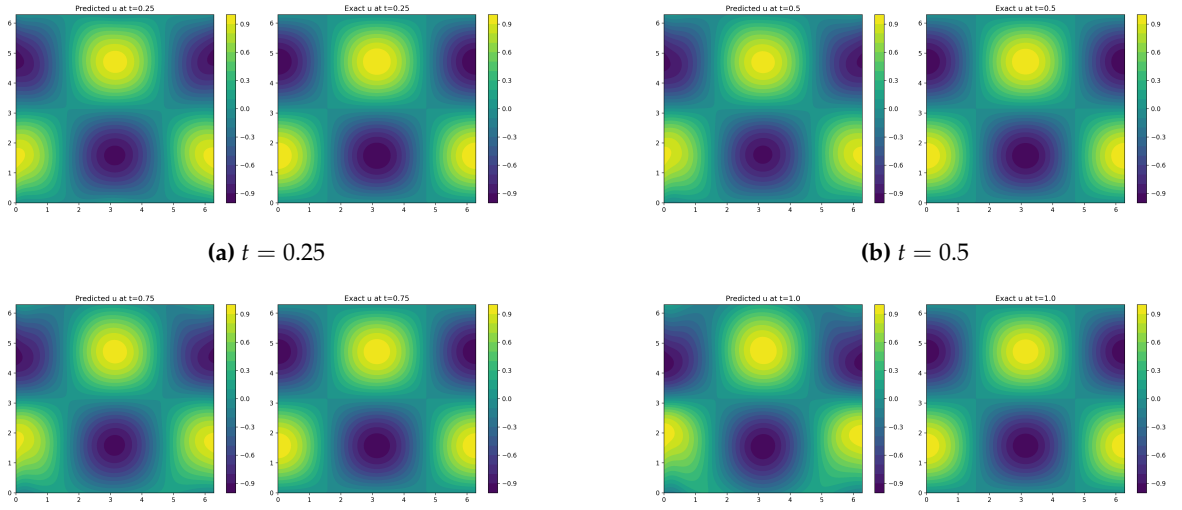
At $t = 0.5$ (Figure 2b), the agreement remains very strong. The vortical structures are still clearly defined, and only minor discrepancies can be observed in regions with steep gradients. These differences are barely noticeable and do not affect the overall flow topology.

At $t = 0.75$ (Figure 2c), small deviations begin to emerge. In particular, the predicted contours exhibit slight smoothing near the vortex cores. This behavior suggests that the PINN tends to under-resolve sharp gradients as time progresses, which is a common feature of neural network approximations when representing increasingly diffused fields.

At the final time $t = 1.0$ (Figure 2d), the smoothing effect becomes more pronounced. The extrema of the velocity field are slightly reduced, and the gradients are less sharp compared to the exact solution. Despite this, the global structure of the flow, including the location of maxima, minima, and nodal lines, is still accurately preserved. This indicates that while local accuracy degrades, the PINN continues to capture the correct large-scale dynamics of the flow.

The Figure 2 demonstrates that the PINN provides highly accurate predictions at early times and maintains reasonable fidelity even at later stages of viscous decay.

 doi:10.20944/preprints202604.1546.v1



(a) $t = 0.25$ **(b)** $t = 0.5$

(c) $t = 0.75$ **(d)** $t = 1.0$

Figure 2. Contours of the horizontal velocity u: predicted (left column) and exact (right column) at increasing times. The PINN reproduces the main features of the decaying vortex, with slight smearing at later times.

To provide a more quantitative assessment, Figure 3 shows the horizontal velocity u along the symmetry line $y = \pi$ for all four time instants. The PINN predictions (solid lines) are compared directly with the exact solution (dashed lines).

At $t = 0.25$, the agreement is nearly perfect across the entire domain. Both the amplitude and phase of the solution are accurately reproduced, indicating that the network has successfully captured the initial condition and early-time dynamics.

At $t = 0.5$, slight deviations begin to appear near the extrema located at $x \approx \pi/2$ and $x \approx 3\pi/2$. These regions correspond to locations where the solution exhibits maximum curvature, making them more difficult for the network to approximate accurately.

As time increases to $t = 0.75$ and $t = 1.0$, the deviations become more noticeable, particularly near the peaks and troughs of the velocity profile. However, the zero-crossing at $x = \pi$ is consistently captured at all times, demonstrating that the global phase accuracy of the solution is preserved.

The observed error growth is consistent with the space–time formulation of PINNs, where approximation errors accumulate over time rather than being controlled through a traditional time-marching scheme. Nevertheless, the deviations remain moderate, confirming the robustness of the method.

3.2. Vorticity Field Analysis

The vorticity field provides a more sensitive measure of solution accuracy, as it involves spatial derivatives of the velocity components. Figure 4 compares the predicted and exact vorticity fields at the same four time instants.

At $t = 0.25$ (Figure 4a), the PINN accurately reproduces the alternating pattern of positive and negative vorticity associated with the counter-rotating vortices. The magnitude and spatial distribution are in excellent agreement with the analytical solution.

At $t = 0.5$ (Figure 4b), the agreement remains strong, although slight smoothing of the peak vorticity values can be observed. This effect is more noticeable than in the velocity field because vorticity depends on spatial derivatives.

At $t = 0.75$ (Figure 4c), the attenuation of peak vorticity becomes more evident. The predicted extrema are lower in magnitude compared to the exact solution, indicating that the network is gradually losing accuracy in capturing sharp rotational features.

At $t = 1.0$ (Figure 4d), the smoothing effect is further amplified. Despite this, the overall structure of the vorticity field, including the location and sign of vortical regions, is still correctly represented.

 doi:10.20944/preprints202604.1546.v1

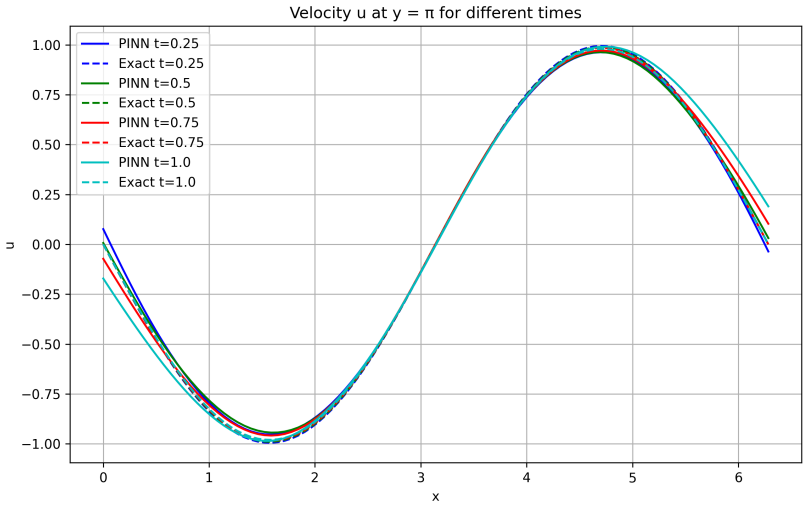
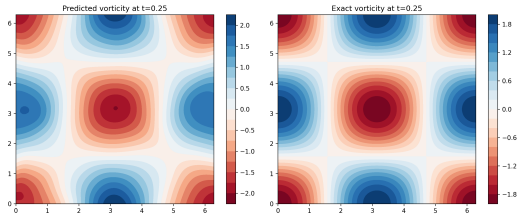
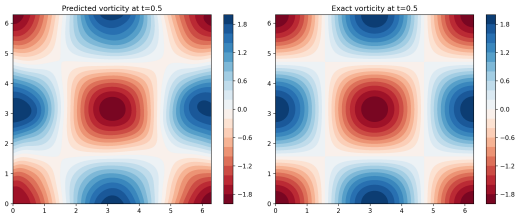


Figure 3. Horizontal velocity u along the line $y = \pi$ for $t = 0.25, 0.5, 0.75, 1.0$. Solid lines are PINN predictions, dashed lines are the exact solution. The PINN follows the exact profile very closely at early times and shows only modest degradation at $t = 1.0$.

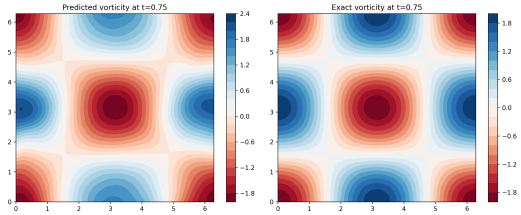
These results highlight an important characteristic of PINNs: while they can accurately capture primary flow features, higher-order quantities such as vorticity are more sensitive to approximation errors.



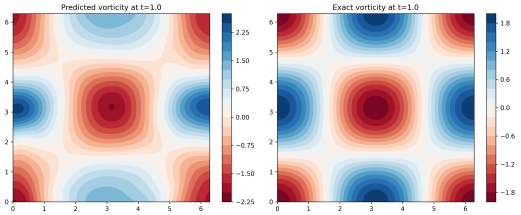
(a) $t = 0.25$



(b) $t = 0.5$



(c) $t = 0.75$



(d) $t = 1.0$

Figure 4. Vorticity contours: predicted (left) and exact (right). The PINN captures the alternating sign and the decay of vorticity over time.

3.3. Three-Dimensional Visualization of Velocity Decay

Figure 5 shows three-dimensional surface plots of the predicted horizontal velocity u at the four time instants. These visualizations provide additional insight into the temporal evolution of the flow.

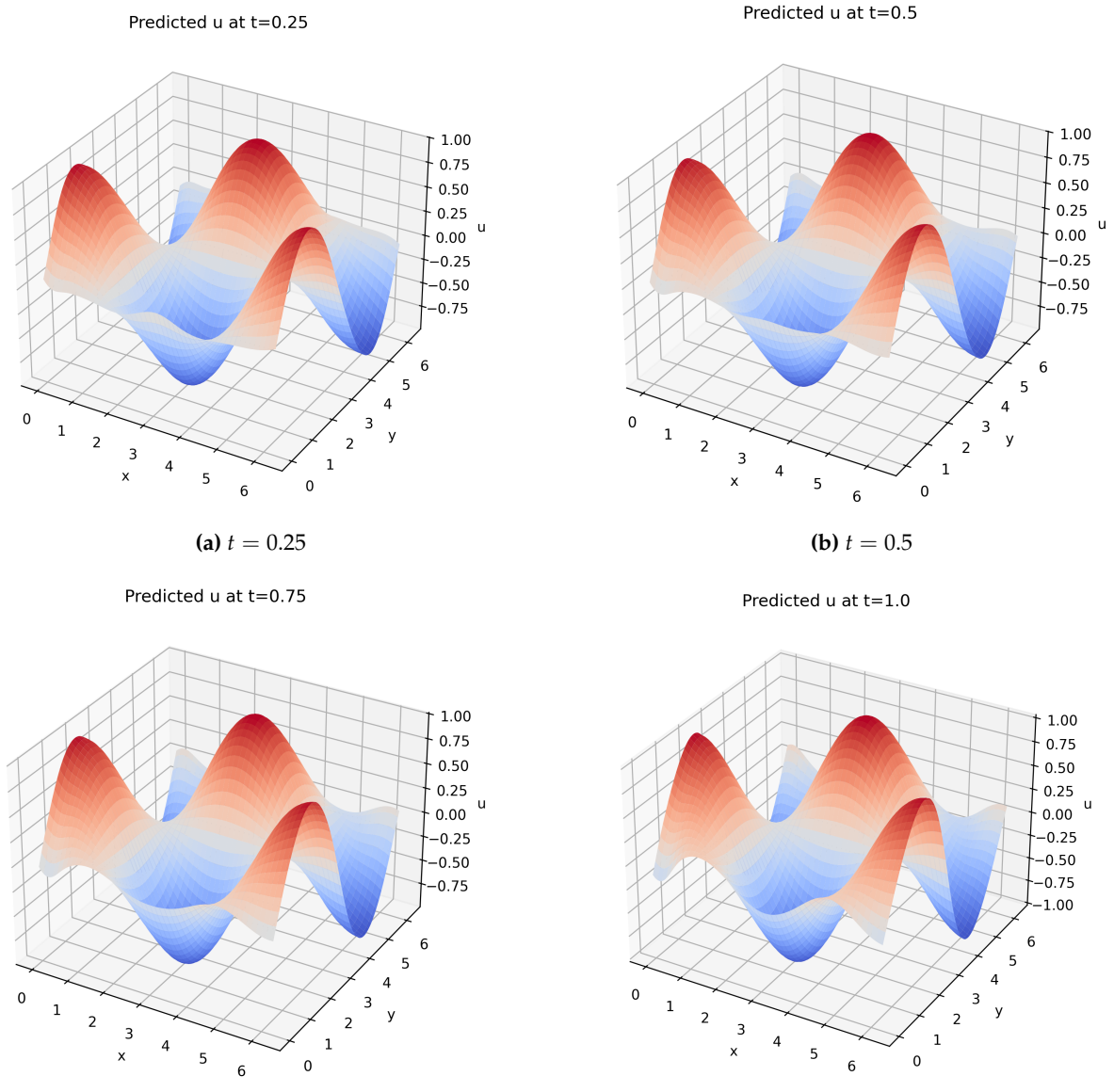
At $t = 0.25$ (Figure 5a), the surface exhibits pronounced peaks and valleys corresponding to strong vortical motion. The sharp gradients indicate high kinetic energy in the flow.

At $t = 0.5$ (Figure 5b), the amplitude of the surface begins to decrease, reflecting the onset of viscous dissipation. The overall shape, however, remains well defined.

 doi:10.20944/preprints202604.1546.v1

At $t = 0.75$ (Figure 5c), the flattening of the surface becomes more apparent. The peaks are reduced in magnitude, and the gradients become smoother, indicating a progressive decay of the vortex strength.

At $t = 1.0$ (Figure 5d), the surface is significantly flattened, consistent with the exponential decay predicted by the analytical solution. Importantly, the surfaces remain smooth and free from spurious oscillations, confirming the stability of the PINN approximation.



(a) $t = 0.25$ **(b)** $t = 0.5$

(c) $t = 0.75$ **(d)** $t = 1.0$

Figure 5. Three-dimensional surfaces of the predicted horizontal velocity u at $t = 0.25, 0.5, 0.75, 1.0$. The gradual flattening of the lobes illustrates the viscous decay of the Taylor–Green vortex.

3.4. Quantitative Error Analysis and Convergence Behavior

Table 1 summarizes the relative L_2 errors for the velocity components and pressure at the selected time instants.

The velocity errors are relatively small at early times, with values around 5% at $t = 0.25$. As time progresses, the errors increase gradually, reaching approximately 17% at $t = 1.0$. This trend reflects the accumulation of approximation errors in the space–time PINN framework.

Due the large magnitude, the pressure error exhibits a decreasing trend over time, suggesting that the network gradually improves its internal consistency in representing the pressure field.

 doi:10.20944/preprints202604.1546.v1

Table 1. Relative L_2 errors for the Taylor–Green vortex at $Re = 100$.

Time t	$L_2(u)$	$L_2(v)$	$L_2(p)$
0.25	5.30e-2	5.79e-2	2.84e0
0.50	6.14e-2	6.74e-2	2.59e0
0.75	9.90e-2	1.05e-1	2.27e0
1.00	1.67e-1	1.79e-1	2.06e0

The training loss history shown in Figure 6 provides further insight into the convergence behavior of the model. A rapid decrease in loss is observed during the first 2 000 iterations, indicating efficient learning of the dominant flow features. Beyond this point, the loss decreases more slowly and eventually reaches a plateau, suggesting that the network has converged to a stable solution.

The final loss value of approximately 3.6×10^{-2} indicates a good balance between the PDE residual and the enforcement of boundary and initial conditions.

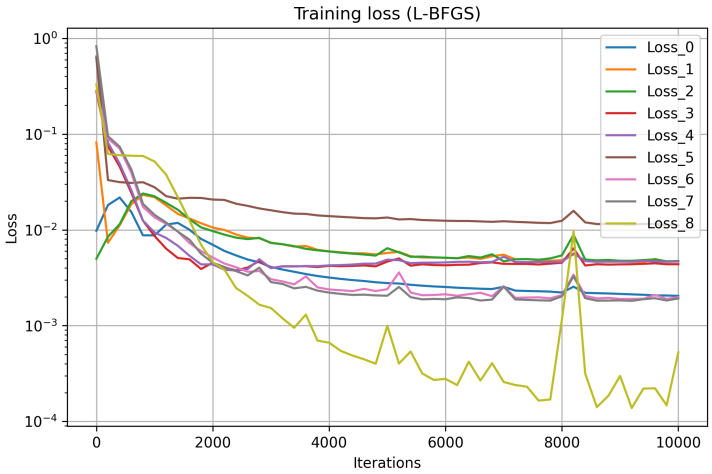


Figure 6. Training loss history (Adam optimizer, 10 000 iterations). The nine curves correspond to the nine components of the loss (continuity, two momentum equations, and six boundary/initial terms). The rapid decrease and subsequent plateau indicate successful convergence.

The results demonstrate that the PINN framework is capable of accurately capturing the evolution of the Taylor–Green vortex, particularly at early times. The method successfully reproduces the main flow structures, including velocity fields and vortex patterns, with reasonable accuracy even at later times.

However, the gradual loss of accuracy observed in both velocity gradients and vorticity highlights a limitation of the approach, especially for long-time simulations. The difficulty in accurately predicting the pressure field further emphasizes the need for improved loss formulations or additional constraints.

Overall, the results indicate that PINNs offer a promising alternative to traditional numerical methods for solving unsteady fluid flow problems, particularly when combined with strategies to improve long-time stability and pressure accuracy.

4. Conclusions

In this work we developed a physics-informed neural network (PINN) for the two-dimensional unsteady Taylor–Green vortex at a Reynolds number of 100, using the DeepXDE library as the implementation framework. The network, which consists of four hidden layers with 128 neurons each and uses the hyperbolic tangent activation function, was trained on 20000 interior collocation points, 2000 boundary points, and 2000 initial points. The Adam optimiser with a learning rate of 10^{-3} was run for 10000 iterations, requiring approximately 13 minutes on a single NVIDIA Tesla T4 GPU.

 doi:10.20944/preprints202604.1546.v1

Our results demonstrate that the PINN is capable of reproducing the main features of the decaying vortex with good accuracy, particularly at early times. The relative L_2 errors for the velocity components u and v increase from about 5.3–5.8% at $t = 0.25$ to about 16.7–17.9% at $t = 1.0$. This gradual error growth is inherent to the space–time formulation of PINNs, where the network learns a global approximation; small deviations introduced at earlier times propagate forward. The vorticity field, which depends on spatial derivatives of the velocity, is also captured qualitatively, although the peak values are smoothed as time increases. Three-dimensional surface plots of the predicted velocity field clearly illustrate the viscous decay of the vortex strength over the considered time interval.

From a practical perspective, the use of DeepXDE greatly simplified the implementation: the geometry, initial and boundary conditions, and the PDE residuals were all specified in a high-level, modular way. The entire pipeline – from data sampling to training and visualisation – was executed in a single Python script, making it highly reproducible. We believe that such reproducible benchmarks are essential for the growing community of researchers applying PINNs to fluid mechanics [19, 24]. Moreover, the success of the method on a moderately complex unsteady flow suggests that PINNs could become a viable alternative to traditional CFD for problems where mesh generation is cumbersome or where only sparse measurements are available [25,26].

However, several limitations of the current approach should be acknowledged. First, we enforced Dirichlet boundary conditions using the exact solution rather than the more natural periodic boundary conditions. This choice was made for implementation convenience, but it means that the network does not explicitly learn periodicity; in problems without a known analytical solution, one would need to implement periodic BCs properly. Second, the training was stopped after 10000 Adam iterations; a subsequent fine-tuning with the L-BFGS optimiser might further reduce the loss, as suggested by [17]. Third, the relatively high pressure errors indicate that the standard PINN loss may be insufficient for problems where pressure is of primary interest.

Future work can address these limitations in several directions.One promising avenue is the use of Fourier feature embeddings, which have been shown to help neural networks learn high-frequency components more effectively [29]. Another direction is residual-based adaptive sampling (RAR), where more collocation points are added in regions of high PDE residual during training [30]. This could improve accuracy near sharp gradients without increasing the total number of points uniformly. For pressure, one could augment the loss function with an explicit pressure-Poisson residual, as done in some recent PINN formulations for incompressible flows [12]. Finally, extending the present framework to higher Reynolds numbers (e.g., $Re = 400$ or 1000) and to the three-dimensional Taylor–Green vortex would provide a more challenging test of the method and help quantify its practical limitations.

So, we have presented a detailed, reproducible PINN study of the unsteady Taylor–Green vortex, including quantitative error tables, multi-time visualisations, and an honest discussion of both successes and difficulties. The code and all generated figures are available from the authors upon request. We hope that this work serves as a useful benchmark for researchers developing and testing PINN methods for fluid dynamics, and that it contributes to the broader effort of making scientific machine learning more transparent and reliable.

Author Contributions: Muhammad Bilal prepared the manuscript and conducted the simulations. Muhammad Sabeel Khan reviewed the manuscript and provided critical feedback.

 doi:10.20944/preprints202604.1546.v1

Nomenclature

Symbol	Description	Unit
u, v	Velocity components in x and y directions	$[m/s]$
p	Pressure (divided by constant density)	$[m^2/s^2]$
x, y	Spatial coordinates	$[m]$
t	Time	$[s]$
ν	Kinematic viscosity	$[m^2/s]$
Re	Reynolds number, $Re = 1/\nu$	$[-]$
T	Final simulation time	$[s]$
Ω	Spatial domain $[0, 2\pi]^2$	$[-]$
$\mathbf{u}$	Velocity vector (u, v)	$[m/s]$
ω	Vorticity, $\omega = \partial v/\partial x - \partial u/\partial y$	$[1/s]$
f_c, f_u, f_v	PDE residuals (continuity, x-momentum, y-momentum)	$[-]$
N_f	Number of interior collocation points	$[-]$
N_b	Number of boundary points	$[-]$
N_0	Number of initial condition points	$[-]$
$\mathcal{L}_{\text{PDE}}$	Loss component for PDE residuals	$[-]$
$\mathcal{L}_{\text{BC}}$	Loss component for boundary conditions	$[-]$
$\mathcal{L}_{\text{IC}}$	Loss component for initial conditions	$[-]$
$\mathcal{L}_{\text{total}}$	Total loss function	$[-]$
$\hat{u}, \hat{v}, \hat{p}$	Neural network predictions for u, v, p	$[m/s], [m/s], [m^2/s^2]$
$\mathbf{z} = (x, y, t)$	Input vector to the neural network	$[m, m, s]$
$L_2(\phi)$	Relative L_2 error for variable ϕ	$[-]$

References

1. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* **2019**, *378*, 686–707.
2. Karniadakis, G.E.; Kevrekidis, I.G.; Lu, L.; Perdikaris, P.; Wang, S.; Yang, L. Physics-informed machine learning. *Nature Reviews Physics* **2021**, *3*, 422–440.
3. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics informed deep learning (part I): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561* **2017**.
4. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics informed deep learning (part II): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566* **2017**.
5. Wang, S.; Yu, X.; Perdikaris, P. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics* **2022**, *449*, 110768.
6. Jagtap, A.D.; Kawaguchi, K.; Karniadakis, G.E. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics* **2020**, *404*, 109136.
7. Shukla, K.; Jagtap, A.D.; Karniadakis, G.E. Parallel physics-informed neural networks via domain decomposition. *Journal of Computational Physics* **2021**, *447*, 110683.
8. Wang, S.; Sankaran, S.; Perdikaris, P. Respecting causality is all you need for training physics-informed neural networks. *arXiv preprint arXiv:2203.07404* **2022**.
9. Cuomo, S.; Di Cola, V.S.; Giampaolo, F.; Rozza, G.; Raissi, M.; Piccialli, F. Scientific machine learning through physics-informed neural networks: Where we are and what's next. *Journal of Scientific Computing* **2022**, *92*, 88.
10. Huang, S.; Feng, W.; Tang, C.; He, Z.; Yu, C.; Lv, J. Partial differential equations meet deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems* **2025**.
11. Mao, Z.; Jagtap, A.D.; Karniadakis, G.E. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering* **2020**, *360*, 112789.
12. Jin, X.; Cai, S.; Li, H.; Karniadakis, G.E. NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics* **2021**, *426*, 109951.

 doi:10.20944/preprints202604.1546.v1

13. Sun, L.; Gao, H.; Pan, S.; Wang, J.X. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering* **2020**, *361*, 112732.
14. Hu, B.; McDaniel, D. Applying physics-informed neural networks to solve Navier–Stokes equations for laminar flow around a particle. *Mathematical and Computational Applications* **2023**, *28*, 102.
15. Cai, S.; Mao, Z.; Wang, Z.; Yin, M.; Karniadakis, G.E. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica* **2021**, *37*, 1727–1738.
16. Faroughi, S.A.; Pawar, N.M.; Fernandes, C.; Raissi, M.; Das, S.; Kalantari, N.K.; Kourosh Mahjour, S. Physics-guided, physics-informed, and physics-encoded neural networks and operators in scientific computing: Fluid and solid mechanics. *Journal of Computing and Information Science in Engineering* **2024**, *24*, 040802.
17. Wang, S.; Teng, Y.; Perdikaris, P. Understanding and mitigating gradient pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing* **2021**, *43*, A3055–A3081.
18. De Ryck, T.; Mishra, S. Error analysis for physics-informed neural networks (PINNs) approximating Kolmogorov PDEs. *Advances in Computational Mathematics* **2022**, *48*, 79.
19. Markidis, S. The old and the new: Can physics-informed deep learning replace traditional linear solvers? *Frontiers in Big Data* **2021**, *4*, 669097.
20. Chuang, P.Y.; Barba, L.A. Experience report of physics-informed neural networks in fluid simulations: pitfalls and frustration. *arXiv preprint arXiv:2205.06751* **2022**.
21. Chuang, P.Y.; Barba, L.A. Predictive limitations of physics-informed neural networks in vortex shedding. *arXiv preprint arXiv:2306.00230* **2023**.
22. Taylor, G.I.; Green, A.E. Mechanism of the production of small eddies from large ones. *Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences* **1937**, *158*, 499–521.
23. Brachet, M.; Meneguzzi, M.; Vincent, A.; Politano, H.; Sulem, P. Numerical evidence of smooth self-similar dynamics and possibility of subsequent collapse for three-dimensional ideal flows. *Physics of Fluids A: Fluid Dynamics* **1992**, *4*, 2845–2854.
24. Lu, L.; Meng, X.; Mao, Z.; Karniadakis, G.E. DeepXDE: A deep learning library for solving differential equations. *SIAM Review* **2021**, *63*, 208–228.
25. Arzani, A.; Wang, J.X.; Sacks, M.S.; Shadden, S.C. Machine learning for cardiovascular biomechanics modeling: challenges and beyond. *Annals of Biomedical Engineering* **2022**, *50*, 615–627.
26. Garay, J.; Dunstan, J.; Uribe, S.; Costabal, F.S. Physics-informed neural networks for blood flow inverse problems. *arXiv preprint arXiv:2308.00927* **2023**.
27. Dong, S.; Li, Z. A general approach for enforcing periodic boundary conditions in physics-informed neural networks. *arXiv preprint arXiv:2108.05991* **2021**.
28. Lu, L. DeepXDE documentation. https://deepxde.readthedocs.io/, 2021.
29. Rahimi, A.; Recht, B. Random features for large-scale kernel machines. In Proceedings of the Advances in Neural Information Processing Systems (NIPS). Curran Associates, Inc., 2007, Vol. 20.
30. Lu, L.; Gao, Y.; Karniadakis, G.E. A comprehensive study of adaptive sampling for physics-informed neural networks. *arXiv preprint arXiv:2106.06003* **2021**.